UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/777,504	02/05/2001	Gregory Robert Roelofs	US010024	8342

7590 09/13/2004

Corporate Patent Counsel
U.S. Philips Corporation
580 White Plains Road
Tarrytown, NY 10591

EXAMINER	
ROSALES HANNER, MORELLA I	

ART UNIT	PAPER NUMBER
2128	

DATE MAILED: 09/13/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/777,504	ROELOFS, GREGORY ROBERT	
	Examiner	Art Unit	
	Morella I Rosales-Hanner	2128	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 February 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 - 20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1 - 20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

Detailed Action

1. **Claims 1 – 20** have been examined and are pending.

Information Disclosure Statement

2. The office acknowledges receive of the information disclosure statement (IDS) submitted on October 23, 2002. The submission is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement has being considered by the examiner.

Specification

3. The disclosure is objected to because it contains an embedded hyperlink and/or other form of browser-executable code **[page 3, line 9]**. Applicant is required to delete the embedded hyperlink and/or other form of browser-executable code. See MPEP § 608.01.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

4.1 **Claims 1 – 20** are rejected under 35 U.S.C. 103(a) as being unpatentable over a printed publication by **David Anderson et al.** entitled "**Tangible Interaction + Graphical Interpretation: A New Approach to 3D Modeling**", hereafter referred to as *Anderson* in view of **U.S Patent No. 5,013,6,262,711 to Cohen et al.**, hereafter referred to as *Cohen*.

4.1.1 As regards to **claims 1**, *Anderson* teaches **[Pg 393, Section 2 Computational Building Blocks]** a system for creating a virtual model, to be displayed on a computer driven display, of a physical structure comprising:

- at least one sensor providing sensor data **[Pg 395, Section 2.1 System Description, last paragraph]**;
- at least one component capable of being sensed by the sensor **[Pg 395, Section 2.1 System Description, first paragraph]**;
- a computer interface for coupling the sensor to a computer, the computer determining the position and dimensions of each component based on the sensor data, and the computer creating a virtual model to be displayed on a

computer display of a structure representative of an arrangement of the components [**Pg 395, Section 2.2 Geometry determination, paragraphs 5 & 6**].

Anderson also teaches [**Pg 395, section 2.2 Geometry determination, paragraph 5**] a special block [component] that supplies power to the blocks, provides an interface between a block structure and a host computer and may be attached to any part of the block structure. *Anderson* further teaches [**Pg 401, section 4 Conclusions and future work, paragraph 3**] that to make a truly useful and affordable system more investigation is required and that current work and plans include reducing the power requirements (and thereby the cost) of the blocks by looking at ways of capturing a block structure using only a bare minimum of active components in each block.

Anderson does not expressly teach a baseboard upon which components are mounted.

Cohen teaches [**Col 6, lines 16 – 46**] a detection field, which is equivalent to Applicant baseboard, that can detect (sense) the position of interactors in "x" and "y" coordinates, it is provided with channels which permit interactors (components) to be engaged (mounted) with the detection field, and it is coupled to a computer system via an interface. *Cohen* also teaches [**Col 2, lines 49 – 52**] that detection field is suitably sized and configured so that multiple users can be engaged simultaneously.

It would have been obvious to one of ordinary skills in the art, at the time of the invention, to modify the system for creating a virtual model and make it more useful and affordable system by reducing the power requirements (and thereby the cost) of the blocks (components) and looking at ways of capturing a block structure

using only a bare minimum of active components in each block, as taught by *Anderson*, by using an apparatus that can detect (sense) the position of multiple blocks (components) simultaneously and that is suitably sized and configured to engage multiple users simultaneously as taught by *Cohen*.

4.1.2 As regards to **claims 2 and 3**, *Anderson* teaches **[section 2.1, System description]** blocks (components) comprising a non conductive material capable of being sensed by the sensor and at least one projecting electrical contact point formed of a conductive material and wherein the sensor comprises a circuit board providing an array of electrical contact holes at predetermined positions, and includes an identification label capable of being sensed by the sensor; wherein the sensor data comprises identification data sensed from the identification label and location and orientation data for each component sensed; and wherein the sensor data is stored by one of the computer and the sensor **[Section 2.2 Geometry determination]**.

4.1.3 As regards to **claim 4**, *Anderson* teaches **[Section 2.2 Geometry determination]** storing property data, representative of the dimensions and shape, associated with the identification data for each block (component).

4.1.4 As regards to **claim 5**, *Anderson* teaches **[Section 2.2 Geometry determination]** identification label of each block (component) comprising an electronic signature; and

the sensor is a circuit board capable of sensing the position of each sensed (mounted) component and its electronic signature.

4.1.5 As regards to **claim 6**, *Anderson* teaches [**Section 2.2 Geometry determination, 4th paragraph**] an identification label for each block (component) wherein the sensor is a circuit capable of reading the position and ID of each connected (mounted) block.

Anderson does not expressly teach an identification label comprising a magnetic signature

Cohen teaches [**Col 6, Line 58 – Col 7, line 8**] an identification label for each interactor (component) comprises a magnetic signature; and wherein the sensor comprises a magnetic sensing board capable of reading the position and magnetic signature of each mounted component. *Cohen* also teaches [**Col 7, Lines 5 – 8**] that magnets not only hold the interactor (components) in position, they also ensure good contact with the surfaces of the interactor.

It would have been obvious to one of ordinary skills in the art, at the time of the invention, to implement the identification label taught by *Anderson* as a magnetic signature in order to hold the blocks in position and to ensure good contact with the surfaces of the interactor (component) as taught by *Cohen*.

4.1.6 As regards to **claim 7**, ,*Anderson* teaches [**Section 2.2 Geometry determination**] sensors on the top detection field of a drain block with the circuit board covered with a nonconductive covering having an array of holes placed at

predetermined positions for exposing an array of electrical contact points on the circuit board.

4.1.8 As regards to **claim 8**, *Anderson* teaches **[Section 2.1 System description]** sensors connected to a power source and accesses a voltmeter for testing for a positive voltage, an ammeter for determining current at contact points having a positive voltage, a switching network and a processor receiving data from the voltmeter and for controlling the voltmeter, ammeter and the switching network.

4.1.9 As regards to **claim 9**, *Anderson* teaches **[Section 2.1 System description]** blocks (components) with two associated electrical contact points, and wherein each electrical contact point of each block (component) comprises a plurality of conductors wherein each of the conductors in one of the electrical contact points is in one to one paired correspondence with one conductor in the associated contact point; wherein each electrical contact hole on the circuit board has a plurality of conductors; and wherein electrical contact between a contact point of a sensed (mounted) block (component) and a contact hole of the circuit board comprises one to one electrical contact between the plurality of conductors in the contact point of the sensed (mounted) block (component) and the plurality of conductors in the contacted contact, hole of the circuit board.

4.1.10 As regards to **claim 10**, *Anderson* teaches [**Section 2.2 Geometry determination**] sensor data comprises the location of contact points on the circuit board having electrical contact with associated contact points of sensed (mounted) blocks (components) and current values read by the ammeter for associated contact points.

4.1.11 As regards to **claim 11**, *Anderson* teaches [**Section 2.1 System description**] paired conductors in each block (component) that are independently electrically connected, each electrical connection comprising at least one resistor selected from a predetermined group of possible resistors; and wherein an identification label of each block (component) is comprised of the selected resistance.

4.1.12 As regards to **claim 12**, *Anderson* teaches [**Section 2.1 System description**] blocks (components) comprising two electrical contacts, each electrical contact comprises three conductors, and each electrical connection between paired conductors comprises one resistor.

4.1.13 As regards to **claims 13 and 15**, *Anderson* teaches [**Section 2.2 Geometry determination**] determining the orientation of each block (component) by determining which of the associated contact points of the block (component) is pulling its top signal line low.

Anderson does not expressly teach using a diode as part of one of the electrical connections in a component.

Cohen teaches **[Fig 5a and corresponding text]** use of a diode as part of an interactor (component) coupled to the contact line to prevent false keying, which is well known to those skilled in the art of keyboard design.

It would have been obvious to one of ordinary skills in the art, at the time of the invention, to modify the block (component) taught by *Anderson* to include a diode coupled to the contact line to prevent false keying as taught by *Cohen.*

4.1.14 As regards to **claim 14**, *Anderson* teaches **[Section 2.1 System description]** a block (component) formed of a nonconductive material, comprising:
a pair of associated electrical contact points; each contact point having a plurality of conductors; each conductor independently connected in one to one correspondence to an associated conductor in the associated contact point with each connection including a resistor selected from a predetermined group of possible resistors; and wherein an electronic signature for identifying the block (component) is comprised of a combination of the resistors of the plurality of the connections of the associated conductors of the block (component).

Anderson further teaches **[Pg 401, section 4 Conclusions and future work, paragraph 3]** that to make a truly useful and affordable system more investigation is required and that current work and plans include reducing the power requirements (and thereby the cost) of the blocks by looking at ways of capturing a block structure using only a bare minimum of active components in each block.

Anderson does not expressly teach a baseboard upon which components are mounted.

Cohen teaches **[Col 6, lines 16 – 46]** a detection field, which is equivalent to Applicant baseboard, that can detect (sense) the position of interactors in "x" and "y" coordinates, it is provided with channels which permit interactors (components) to be engaged (mounted) with the detection field, and it is coupled to a computer system via an interface. *Cohen* also teaches **[Col 2, lines 49 – 52]** that detection field is suitably sized and configured so that multiple users can be engaged simultaneously.

It would have been obvious to one of ordinary skills in the art, at the time of the invention, to modify the system for creating a virtual model and make it more useful and affordable system by reducing the power requirements (and thereby the cost) of the blocks (components) and looking at ways of capturing a block structure using only a bare minimum of active components in each block, as taught by *Anderson*, by using an apparatus that can detect (sense) the position of multiple blocks (components) simultaneously and that is suitably sized and configured to engage multiple users simultaneously as taught by *Cohen.*

4.1.16 As regards to **claim 16**, *Anderson* teaches **[Section 2.2 Geometry determination]** a software application receiving sensor data for at least one block (component), comprising computer program code; wherein the sensor data comprises data representative of an identity, orientation and a location for each block (component) sensed (mounted); wherein the computer code processes the sensor data for

determining the identity, position and orientation of each block (component) sensed; wherein the computer code accesses property data including data representative of the dimensions and shape of each block (component) available for sensing, for determining the dimensions and shape of each block (component) sensed in accordance with the identity of each block; the computer code creating a virtual image representative of an arrangement of the blocks (components) sensed (mounted) on the drain block based on the shape, dimensions, orientation and location of each block (component) sensed (mounted) by the drain block.

Anderson does not expressly teach a baseboard upon which components are mounted.

Cohen teaches **[Col 6, lines 16 – 46]** a detection field, which is equivalent to Applicant baseboard, that can detect (sense) the position of interactors in "x" and "y" coordinates, it is provided with channels which permit interactors (components) to be engaged (mounted) with the detection field, and it is coupled to a computer system via an interface. *Cohen* also teaches **[Col 2, lines 49 – 52]** that detection field is suitably sized and configured so that multiple users can be engaged simultaneously.

It would have been obvious to one of ordinary skills in the art, at the time of the invention, to modify the system for creating a virtual model and make it more useful and affordable system by reducing the power requirements (and thereby the cost) of the blocks (components) and looking at ways of capturing a block structure using only a bare minimum of active components in each block, as taught by *Anderson*, by using an apparatus that can detect (sense) the position of multiple blocks

(components) simultaneously and that is suitably sized and configured to engage multiple users simultaneously as taught by *Cohen*.

4.1.17 As regards to **claim 17**, *Anderson* teaches **[Section 2.1 System specification]** a drain block that comprises a circuit board; wherein the sensor data comprises data indicative of current values and associated resistance associated with each contact point of a grid of contact points on the circuit board; wherein the identity is determined according to the resistance associated with each interactor (component) mounted on the circuit board.

4.1.18 As regards to **claim 18**, *Anderson* teaches **[Section 2.2 Geometry determination]** a sensor for sensing the identity, location and orientation of blocks (components) sensed by a drain block comprising;
a circuit board, mounted on the drain block, having a grid of contact points, each contact point having a plurality of conductors; wherein the sensor is connected to a power source and accesses a voltmeter, an ammeter and a switching network; and wherein the sensor further accesses a processor for receiving data from the voltmeter and for controlling the voltmeter, ammeter and the switching network.

Anderson does not expressly teach a baseboard upon which components are mounted.

Cohen teaches **[Col 6, lines 16 – 46]** a detection field, which is equivalent to Applicant baseboard, that can detect (sense) the position of interactors in "x" and "y"

coordinates, it is provided with channels which permit interactors (components) to be engaged (mounted) with the detection field, and it is coupled to a computer system via an interface. *Cohen* also teaches [**Col 2, lines 49 – 52**] that detection field is suitably sized and configured so that multiple users can be engaged simultaneously.

It would have been obvious to one of ordinary skills in the art, at the time of the invention, to modify the system for creating a virtual model and make it more useful and affordable system by reducing the power requirements (and thereby the cost) of the blocks (components) and looking at ways of capturing a block structure using only a bare minimum of active components in each block, as taught by *Anderson*, by using an apparatus that can detect (sense) the position of multiple blocks (components) simultaneously and that is suitably sized and configured to engage multiple users simultaneously as taught by *Cohen.*

4.1.19 As regards to **claim 19**, *Anderson* teaches [**Section 2.2 Geometry determination**] a method for creating a virtual model, to be displayed on a computer driven display, representative of at least one block (component) mounted on a drain block,

wherein each block mounted on the drain block makes electrical contact with an electrical circuit board formed on the baseboard and wherein the circuit board has an array of contact points, each contact point having a predetermined location on the circuit board; comprising the steps of:

- successively applying a high impedance voltage to each contact point for testing each contact point of the array of contact points, for determining the presence and location of a block in electrical contact with the contact point being tested **[Section 2.2 Geometry determination]**;
- measuring the voltage for contact points on the circuit board within a predetermined radius of the contact point being tested **[Section 2.2 Geometry determination]**;
- determining that a block is in electrical contact with the test contact point and an associated contact point having a nonzero measured voltage, at the locations of the contact point being tested and the associated contact point, wherein the location of the contact point being tested and the associated contact point is location data for the block determined to be in electrical contact **[Section 2.2 Geometry determination]**;
- applying a low impedance voltage to the contact point being tested when determined to be in electrical contact with a mounted block **[Section 2.2 Geometry determination]**;
- sensing the current values for the contact point being tested and its associated contact point indicative of an identification of the block determined to be in contact with the contact point being tested and its associated contact point, wherein the identification of the block is identification data **[Section 2.2 Geometry determination]**;

- creating a virtual model representative of an arrangement of the blocks when connected to the drain block according to a structure composed of each of the block based on the location data, identification data and property data for each block **[Section 2.2 Geometry determination]**.

- consulting a database of block identifications storing property data comprising dimension data for each block identification; and

Anderson further teaches **[Pg 401, section 4 Conclusions and future work, paragraph 3]** that to make a truly useful and affordable system more investigation is required and that current work and plans include reducing the power requirements (and thereby the cost) of the blocks by looking at ways of capturing a block structure using only a bare minimum of active components in each block.

Anderson does not expressly teach a baseboard upon which components are mounted or consulting a database of block identifications storing data comprising dimension data for each block identification.

Cohen teaches **[Col 6, lines 16 – 46]** a detection field, which is equivalent to Applicant baseboard, that can detect (sense) the position of interactors in "x" and "y" coordinates, it is provided with channels which permit interactors (components) to be engaged (mounted) with the detection field, and it is coupled to a computer system via an interface. *Cohen* also teaches **[Col 2, lines 49 – 52]** that detection field is suitably sized and configured so that multiple users can be engaged simultaneously. *Cohen* further teaches **[Fig 16, step 294 and corresponding text]** consulting a database of interactor

identifications storing property data comprising dimension data for each interactor identification as part of detecting privacy violation.

It would have been obvious to one of ordinary skills in the art, at the time of the invention, to modify the system for creating a virtual model and make it more useful and affordable system by reducing the power requirements (and thereby the cost) of the blocks (components) and looking at ways of capturing a block structure using only a bare minimum of active components in each block, as taught by *Anderson*, by using an apparatus that can detect (sense) the position of multiple blocks (components) simultaneously and that is suitably sized and configured to engage multiple users simultaneously as taught by *Cohen* and to consult a database of interactor (component) identification to detect privacy violation as also taught by *Cohen.*

4.1.20 As regards to **claim 20**, *Anderson* teaches **[Section 2.2 Geometry determination]** creating a virtual model representative of an arrangement of the blocks when connected to the drain block according to a structure composed of each of the block based on the location data, identification data and property data for each block using a straightforward recursive procedure.

Additional references

5. The following is a list of references that are relevant to the claimed invention but were not cited by the examiner:

- Matthias Rauterberg, Morten Fjeld, Helmut Krueger, Martin Bichsel, Uwe Leonhardt, Markus Meier ; "BUILD-IT: a planning tool for construction and design"; April 1998; CHI 98 conference summary on Human factors in computing systems
- David Anderson, James L. Frankel, Joe Marks, Darren Leigh, Eddie Sullivan, Jonathan Yedidia, Kathy Ryall; "Building virtual structures with physical blocks"; Nov 1999; Proceedings of the 12th annual ACM symposium on User interface software and technology
- George W. Fitzmaurice, Hiroshi Ishii, and William Buxton; "Bricks:Laying the Foundations for graspable User Interfaces", CHI '95 Proceedings
- Matthew G. Gorbet, Maggie Orth, Hiroshi Ishii; "Triangles: tangible interface for manipulation and exploration of digital information topography"; Jan 1998; Proceedings of the SIGCHI conference on Human factors in computing systems
- Brygg Ullmer, Hiroshi Ishii, Dylan Glas; "mediaBlocks: physical containers, transports, and controls for online media";July 1998; Proceedings of the 25th annual conference on Computer graphics and interactive techniques

6. Any inquiry concerning this communication or earlier communication from the examiner should be directed to Morella Rosales-Hanner whose telephone number is

(703) 305-8883. The examiner can normally be reached Monday-Friday from 7:00 a.m. to 3:30 p.m.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jean Homere can be reached on 703 308-6647. The fax number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

MRH

~~Aug 24th, 2004~~

JEAN R. HOMERE
PRIMARY EXAMINER

9/7/04

#3 C Burns 5-23-01

Form PTO-1449 (REV. 7-80) U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	Atty. Docket No. US010024	Serial No.
	Applicant Greg Roelofs	
INFORMATION DISCLOSURE CITATION (Use several sheets if necessary)	Filing Date CONCURRENTLY	Group

JC918 U.S. PTO 09/77504 02/06/01

U.S. PATENT DOCUMENTS

Ex. Int.		Document Number	Date	Name	Class	Sub-class	Filing Date If Approp.
mel	AA	5 7 6 8 1 3 4	06/16/98	Swaelens et al.	364	468.28	4/11/95
mel	AB	5 8 1 3 9 8 4	09/29/98	Haaga et al.	600	410	3/7/97
	AC						
	AD						
	AE						
	AF						

FOREIGN PATENT DOCUMENTS

		Document Number	Date	Country	Class	Sub-class	Trans. Yes	Trans. No
mel	AG	2 7 6 0 1 1 9	02/24/97	France	G09B	1/02		X
mel	AH	W O 9 7 2 3 8 4 5	07/03/97	PCT	G06T	17/00	X	
mel	AI	1 0 3 3 6 7 9	09/06/00	EP	G06T	1/00	X	
	AJ							
	AK							

OTHER (Including Author, Title, Date, Pertinent Pages, Etc.)

mel	AL	Advid Anderson, et al., Tangible Interaction + Graphical Interpretation: A New Approach to 3D Modeling, In Proceedings of SIGGRAPH 2000, July 23-28, 2000. New Orleans, LA
	AM	
	AN	

Examiner [signature] Date Considered 08/24/2004

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to applicant.

Please type a plus sign (+) inside this box → [+]

PTO/SB/08A (08-00) - MODIFIED
Approved for use through 10/31/2002. OMB 0651-0031
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet 1 of 1

Application Number	09/777,504
Filing Date	February 5, 2001
First Named Inventor	Greogory Robert Roelofs
Group Art Unit	2123
Examiner Name	
Attorney Docket Number	US010024

U.S. PATENT DOCUMENTS

Examiner* Initials	Cite No.[1]	U.S. Patent Document Number	Kind Code[2] (if known)	Name of Patentee or Applicant of Cited Document	Date of Publication MM-DD-YYYY	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	1.					

FOREIGN PATENT DOCUMENTS

Examiner* Initials	Cite No.[1]	Foreign Patent Document Office[3]	Number[4]	Kind Code (if known)[5]	Name of Patentee or Applicant of Cited Document	Date of Publication MM-DD-YYYY	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
MRH			DE3602467	✓	SCHWAB Technologieberatung	07-30-1987		
MRH			WO9706479	✓	Royal College of Art; Interval Research Corp (US)	02-20-1997		

NON PATENT LITERATURE DOCUMENTS

Examiner* Initials	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[6]
MRH		MARTIN, F. AND BOROVOY, R.: "THE ACTIVE LEGO BASEPLATE PROJECT" INTERNET ARTICLE, ONLINE 1994, XP002203961, RETRIEVED FROM INTERNET: <URL:fredm.www.media.mit.edu/people/fredm/projects/ab> RETRIEVED ON 2002-06-26	

Examiner Signature		Date Considered	05/24/2004

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] Unique citation designation number. [2] See attached Kinds of U.S. Patent Documents. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

Notice of References Cited	Application/Control No. 09/777,504	Applicant(s)/Patent Under Reexamination ROELOFS, GREGORY ROBERT	
	Examiner Morella I Rosales-Hanner	Art Unit 2128	Page 1 of 2

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,262,711	07-2001	Cohen et al.	345/156
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Matthias Rauterberg, Morten Fjeld, Helmut Krueger, Martin Bichsel, Uwe Leonhardt, Markus Meier ;"BUILD-IT: a planning tool for construction and design"; Apr 1998; CHI 98 conference summary on Human factors in computing systems□□
	V	David Anderson, James L. Frankel, Joe Marks, Darren Leigh, Eddie Sullivan, Jonathan Yedidia, Kathy Ryall; "Building virtual structures with physical blocks"; Nov 1999; Proceedings of the 12th annual ACM symposium on User interface software and technology
	W	George W. Fitzmaurice, Hiroshi Ishii, and William Buxton; "Bricks:Laying the Foundations for graspable User Interfaces", CHI '95 Proceedings
	X	Matthew G. Gorbet, Maggie Orth, Hiroshi Ishii; "Triangles: tangible interface for manipulation and exploration of digital information topography"; Jan 1998; Procs. of the SIGCHI conference on Human factors in computing systems □□

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Notice of References Cited	Application/Control No. 09/777,504	Applicant(s)/Patent Under Reexamination ROELOFS, GREGORY ROBERT	
	Examiner Morella I Rosales-Hanner	Art Unit 2128	Page 2 of 2

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Brygg Ullmer, Hiroshi Ishii, Dylan Glas; "mediaBlocks: physical containers, transports, and controls for online media";Jul 1998; Procs of the 25th annual conference on Computer graphics and interactive techniques □□
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Building Virtual Structures With Physical Blocks

David Anderson[2], James L. Frankel[1,2], Joe Marks[2],
Darren Leigh[2], Kathy Ryall[3], Eddie Sullivan[2], Jonathan Yedidia[2]

[1]Frankel and Associates, Inc., Lexington, MA 02421
[2]MERL — A Mitsubishi Electric Research Laboratory, Cambridge, MA 02139
[3]University of Virginia, Dept. of Computer Science, Charlottesville, VA 22903

ABSTRACT

We describe a tangible interface for building virtual structures using physical building blocks. We demonstrate two applications of our system. In one version, the blocks are used to construct geometric models of objects and structures for a popular game, Quake II™. In another version, buildings created with our blocks are rendered in different styles, using intelligent decoration of the building model.

KEYWORDS: Tangible user interfaces, transmedia.

OVERVIEW

Few people know how to use graphics modeling packages, but everyone can build things out of blocks. Starting from this premise, and with the long-term goal of developing an accessible modeling tool for building virtual worlds, we developed a novel object-modeling system comprising building blocks that self-describe the geometric structures into which they are assembled. Each building block contains a microcontroller, and is able to communicate with the blocks to which it is physically connected. The blocks in an assembled structure use a distributed algorithm to first discover how they are connected to their immediate neighbors. This information is then relayed from block to block — each of our block structures is essentially a self-configuring, store-and-forward computer network — until it reaches the host computer. From the block connectivity data that it collects, and knowledge of the shape of each block, the host computer recovers the geometric structure of the assembled blocks. The structure is then rendered in various styles, ranging from a literal rendition in which blocks look like Lego™ bricks, to decorative interpretations in which structural elements are identified automatically and decorated appropriately. Once described, the virtual structures are available for viewing by the user. Sensors and transducers in the blocks also allow the physical structure to serve as an I/O device for interacting with the virtual structure. Fig. 1a-e shows a physical block structure and sample renderings of the virtual model recovered from it.

UIST '99. Asheville, NC

RELATED WORK

The idea of a self-describing construction kit is not new: several groups have developed such systems over the past 20 years, beginning with the pioneering work of Aish and Frazer [1, 3], and continuing with more recent projects [2, 4].

Our self-describing building blocks resemble several previous systems, but differ in the following key ways:

- *Lego™-like block design:* Basing the physical design of our blocks on the ever-popular Lego™ brick has two distinct advantages. One is the structural rigidity and geometric constraint inherent in the design, which facilitates the recovery of 3D geometry from connectivity data. The more significant advantage is configurability: our blocks can be arranged in geometric structures of greater variety.
- *Minimalist physical/electrical connection:* To make our blocks more configurable requires the use of simple, symmetric connectors. Our block has eight plugs on top, and eight jacks on the bottom. The plugs and jacks have only two conductors each, one for power distribution and one for bidirectional signals.
- *Asynchronous, distributed communication:* Simple connectors lead to a more complicated communication architecture. In particular, our blocks do not enjoy the advantage of a common bus, which greatly simplified some of the previous systems. All communication in our block structures is based on asynchronous message passing between physically connected blocks.

TECHNICAL DETAILS

A block consists of a 100mm (L) × 50mm (W) × 25mm (H) plastic box that is drilled to accommodate slightly modified DC power connectors. The dimensions of the box and the locations of the plugs and jacks are such that our building blocks can be configured like Lego™ bricks. The connectors are mounted on a PC board that also accomodates a microcontroller (a PIC16C77), various passive components, and optional transducers and sensors (Fig. 1f).

Each connector has just two conductors. However, instead of using one for power and one for ground (normal usage for such a connector), we use the inner pin as a signal line for bidirectional communication, and the outer sleeve for power distribution. Each block is wired internally so that connector sleeves carrying power and ground, respectively, are ar-

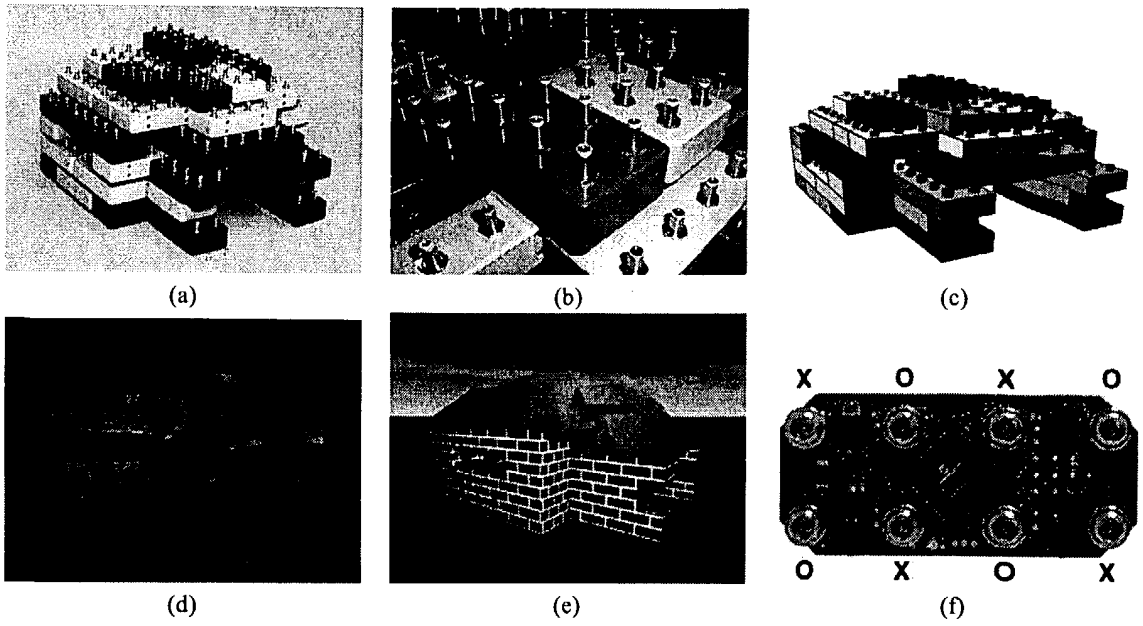

Figure 1: A physical block structure comprising 50 blocks (a), a close-up of the structure (b), renderings of the virtual model computed from it, two literal (c,d), and one more decorative (e). The literal renderings use preassigned shapes and colors to render the blocks. The virtual model is parsed and augmented automatically for the decorative rendering. A bottom view of the printed circuit board inside each block (f) shows eight connector jacks, four LEDs, and the microcontroller. The polarity of the 'X' connector sleeves is different from that of the 'O' sleeves.

ranged in an alternating pattern (Fig. 1f). Thus each block will have at least one connection to power and one to ground in any typical Lego™-like structure. A full-wave bridge rectifier copes with this ambiguity.

A fully assembled block structure computes its own geometry in three phases. In Phase 1, each block determines the pins with which it connects to some other block. In Phase 2, connected blocks exchange block and pin ID data over their connected pins. During these two phases, all blocks compute and communicate in parallel, using power-on for initial (but approximate) synchronization. In Phase 3, the connectivity data collected by each block in Phase 2 are communicated to a host computer via the *drain*, a special block that can be attached to any part of a block structure, and that supplies power to the structure and a serial connection to the host computer. Phase 3's information-draining operation is performed by a distributed, depth-first traversal of the block structure. This operation is serial, but the relay of data packets from block to block back to the drain is pipelined, thus achieving some parallelism in Phase 3, too.

At the end of Phase 3, the host computer has complete connectivity information for the block structure. The host also has shape and appearance data for each block, indexed by block ID, and recorded when a block is programmed. A simple recursive procedure now gives the 3D geometry of the block structure, which is used to produce a scene description suitable for a variety of 3D graphics applications (Fig. 1c,d).

Our blocks might appear too coarse in scale and shape to build richly decorated virtual structures. However, even a limited ability to parse a structure allows the host computer to perform detailing tasks that would be tedious for the user. Our system generates a description of a block structure as a set of logical axioms, one to assert the existence and location of each block. These axioms serve as input to a Prolog program that identifies structural elements of a block structure, interpreted as a building of some kind. Recognized structural elements are then colored and textured distinctively, or decorated with additional geometry, to automatically enhance the visual appearance of the rendered model (Fig. 1e).

REFERENCES

1. R. R. Aish. Modelling arrangements, U.S. patent #4,275,449, 1981.

2. G. Anagnostou, D. Dewey, and A. Patera. Geometry-defining processors for engineering design and analysis. *The Visual Computer*, 5:304–315, 1989.

3. J. Frazer. *An Evolutionary Architecture*. Architectural Association, London, 1994. Includes accounts of several self-describing construction kits developed by Frazer's group from 1981 onwards.

4. M. G. Gorbet, M. Orth, and H. Ishii. Triangles: Tangible interface for manipulation and exploration of digital information topography. In *Proc. of CHI 98*, pages 49–56, Los Angeles, CA, Apr. 1998. ACM.

mediaBlocks: Physical Containers, Transports, and Controls for Online Media

Brygg Ullmer, Hiroshi Ishii, and Dylan Glas*
Tangible Media Group
MIT Media Lab

Abstract

We present a tangible user interface based upon *mediaBlocks:* small, electronically tagged wooden blocks that serve as physical icons ("phicons") for the containment, transport, and manipulation of online media. MediaBlocks interface with media input and output devices such as video cameras and projectors, allowing digital media to be rapidly "copied" from a media source and "pasted" into a media display. MediaBlocks are also compatible with traditional GUIs, providing seamless gateways between tangible and graphical interfaces. Finally, mediaBlocks act as physical "controls" in tangible interfaces for tasks such as sequencing collections of media elements.

CR Categories and Subject Descriptors: H.5.2 [User Interfaces] Input devices and strategies; H.5.1 [Multimedia Information Systems] Artificial, augmented, and virtual realities

Additional Keywords: tangible user interface, tangible bits, phicons, physical constraints, ubiquitous computing

1 INTRODUCTION

Computers have traditionally recorded digital information on both "fixed" and "removable" storage media. While removable media have often been limited in capacity, speed, and accessibility, these factors have been offset by the expanded storage and mobility removable media affords.

However, the rise of widespread online connectivity for both computers and other digital media devices (cameras, projectors, etc.) alters this historical role division. Extended capacity and instant mobility are now better afforded by keeping media online. Does removable media risk obsolescence in the online age?

From a user interface standpoint, the process of online media exchange between digital whiteboards, projectors, computers, and other devices is still far from seamless. Reference and manipulation of online media is at present generally limited to GUI-based interaction with file paths, URLs, and hyperlinks – a process quite at odds with most media interfaces.

We believe that coupling the physicality of removable media with the connectivity and unlimited capacity of online content offers a potential solution to this problem. Moreover, we believe this approach suggests new physical-world interface possibilities that go beyond the pervasive graphical user interface.

*{ullmer,ishii,krill}@media.mit.edu
20 Ames St., E15-445, Cambridge, MA 02139
http://tangible.media.mit.edu/

In this paper, we introduce a tangible user interface (TUI) based upon *mediaBlocks**: small wooden blocks that serve as physical icons ("phicons") [15] for the containment, transport, and manipulation of online media. MediaBlocks do not actually store media internally. Instead, they are embedded with ID tags that allow them to function as "containers" for online content, or alternately expressed, as a kind of physically embodied URL.

MediaBlocks interface with media input and output devices such as video cameras and projectors, allowing digital media to be rapidly "copied" from a media source and "pasted" into a media display. MediaBlocks are also compatible with traditional GUIs, providing seamless gateways between tangible and graphical interfaces. Finally, mediaBlocks are used as physical "controls" in tangible interfaces for tasks such as sequencing collections of media elements.

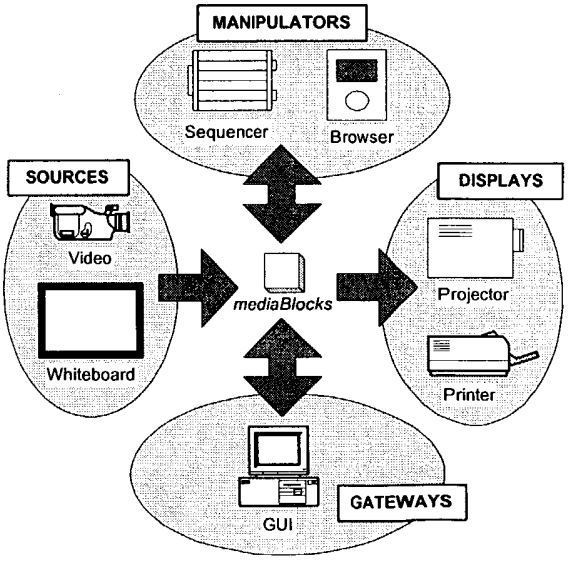

Figure 1: mediaBlocks design space

The mediaBlocks design space is illustrated in Figure 1. MediaBlocks serve as a medium of interchange between media source and display devices; between media devices and GUI-based computers; and between these pre-existing devices and new tangible interfaces for media manipulation. In this fashion, mediaBlocks fill the user interface gap between physical devices, digital media, and online content.

*Note: Our mediaBlocks have no relation to the Magnifi Inc. software product of the same name.

2 FUNCTIONALITY OVERVIEW

The following paragraphs describe the basic functionality of our mediaBlocks interfaces. Detailed consideration of interface use and implementation follows later in the paper.

2.1 Physical Containers

MediaBlocks are phicons (physical icons, [15]) embodied as small wooden blocks. MediaBlocks do not actually store digital media internally. Instead, they physically contain ID tags that are dynamically associated with sequences of online media elements. When used in environments where many media devices are linked to online computation, mediaBlocks act as physical "containers" for online media.

As such, mediaBlocks have a variety of interesting properties. Because contents remains online, mediaBlocks have unlimited "capacity" and rapid transfer speed (copying is instantaneous, while playback is a function of network bandwidth). For the same reason, a lost block is easily replaced. MediaBlocks may also contain live streaming media.

2.2 Physical Transports

One role of mediaBlocks is support for simple physical transport and interchange of media between media devices. While inter-application "copy and paste" is core to the modern GUI, comparably lightweight equivalents have not existed for physical media devices. We have realized a physical analog of "copy and paste" by combining mediaBlocks with physical slots mounted upon associated media devices.

We have implemented mediaBlock support for four media devices: a desktop video projector, network printer, video camera, and digital whiteboard. Inserting a block into the slot of a media source begins recording to an online server. Recording stops when the block is removed. This can be understood as "copying" from the media source into the block. Similarly, contents may be "pasted" into a media display by inserting a block into the associated slot. This will display block contents, with removal halting playback.

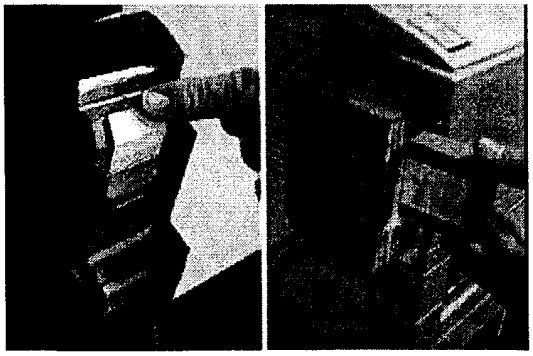

Figure 2: Whiteboard, printer mediaBlock slots

2.3 Physical Gateways

MediaBlock slots have also been implemented for use on general-purpose computers. Slots are mounted on the right margins of computer monitors. When a mediaBlock is inserted into the slot, a GUI window scrolls "out of the block" from the slot's left edge (contiguous with the screen). This window provides GUI access to block contents. (Figure 3)

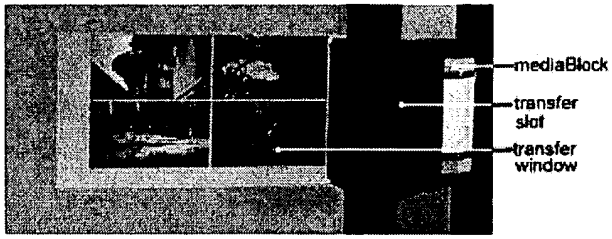

Figure 3: mediaBlock monitor slot

MediaBlock contents may then be transferred to the desktop or to GUI applications with conventional mouse-based "drag and drop" support. Media may also be copied into blocks in this fashion. In this way, mediaBlocks can be used to seamlessly exchange content between computers and media sources, displays, or other computers.

2.4 Physical Browsers

While the transport function of mediaBlocks allows media to be exchanged between various output devices, it does not address interactive control of media playback, especially for mediaBlocks containing multiple media elements. The *media browser* is a simple tangible interface for navigating sequences of media elements stored in mediaBlocks. (Figure 4)

The browser is composed of a detented browse wheel, video monitor, and mediaBlock slot. Useful both in casual viewing and formal presentation contexts, the browser supports the interactive navigation of mediaBlocks sequences for projector-based display, as well as displaying media on its local screen.

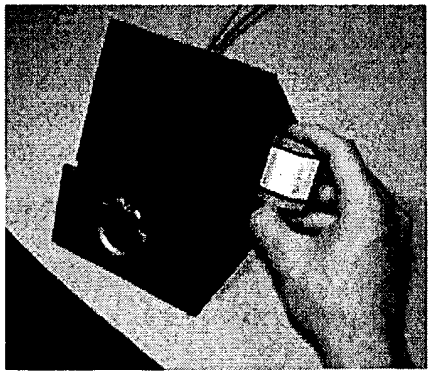

Figure 4: Media browser device

2.5 Physical Sequencers

The media browser provides interactive physical control of mediaBlock display, but does not support modification of mediaBlock contents. The *media sequencer* is a tangible interface using mediaBlocks both as containers and *controls* for physically sequencing media elements. (Figure 5)

Where earlier sections have introduced mediaBlock slots, the sequencer uses physical *racks*, *stacks*, *chutes*, and *pads* as structures that physically and digitally operate upon media-

Blocks. In particular, the rack is a *physical constraint* used to digitally index and sequence mediaBlock contents as a function of the blocks' physical configuration on the rack.

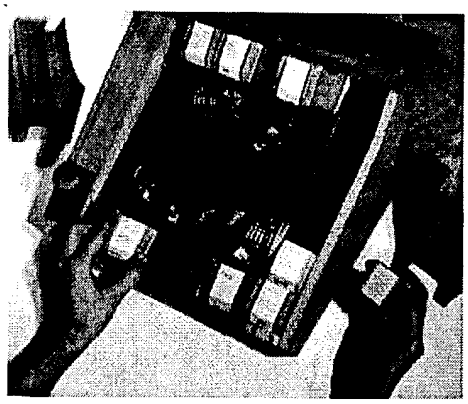

Figure 5: Media sequencer device

2.6 Integrated Functionality

In addition to illustrating the above mediaBlock functions, the accompanying video figure demonstrates the creation of a multimedia presentation integrating all behaviors we have described.

In less than four minutes of uncut footage, we show recording and transport of digital video and whiteboard content; selecting photographs and authoring textual slides for inclusion; assembling these contents into an integrated presentation; rendering the presentation to a network printer; and presenting this content on the browser-controlled video projector. The resulting content is also Web-accessible at each stage of authoring and delivery.

We believe this interface example is significant for several reasons. First, this example shows the creation, manipulation, and use of complex multimedia content with a simplicity and speed that we believe to be highly competitive with other approaches. Simultaneously, it is key to note that the only keyboard, mouse, or other GUI interaction present in the entire sequence is the composition of a textual slide.

In the remainder of the paper, we will continue to develop the interface process, technical operation, and functional roles that we believe represent a powerful new approach for interaction between people, physical objects, and online digital information.

3 RELATED WORK

The mediaBlocks project was most directly influenced by the metaDESK/Tangible Geospace prototype [15, 8] that introduced the phicon concept. Tangible Geospace was based upon an augmented physical desktop and phicons representing geographical landmarks. Manipulation of phicons controlled the position, rotation, and scaling of spatially coincident graphical landscapes.

Tangible Geospace was developed in part to explore physical instantiation of the GUI metaphor. As the GUI system of windows, controls, and icons itself drew from a metaphor of the physical desktop, the physical analogs of lenses, instruments, and phicons (physical icons) seemed a promising first step towards the realization of tangible interfaces.

In practice, Tangible Geospace served to illustrate a number of major differences between graphical and tangible UIs. The fundamental malleability and extent of control over the GUI's graphical workspace differs substantially from the object persistence and cumulative, potentially inconsistent physical degrees of freedom expressed by TUI elements. In short, the GUI metaphor appeared unable to generalize across the potential design space of tangible user interfaces.

Subsequent research with Triangles and inTouch made a key insight towards resolving this shortcoming [7, 2]. Instead of relying upon pre-existing metaphors of GUI or the physical world (e.g., the metaDESK's optical metaphor [8]), these projects developed new interface metaphors based on the affordances unique to physical/digital artifacts. Both projects served as partial inspiration for mediaBlocks, which continues this design aesthetic.

The whiteboard-based mediaBlock functionality draws upon an earlier whiteboard TUI called the transBOARD [8]. The transBOARD used paper cards called *hypercards* as physical carriers for live and recorded whiteboard sessions. However, the hypercard interaction relied upon barcode wanding, which was found cumbersome in practice.

The ubiquitous computing vision of [16] speaks to moving computation and networking off the desktop and into many devices occupying niche contexts within the physical environment. This insight is core to the mediaBlocks system. Still, the interface prototypes of [16] continued to rely primarily upon GUI-based approaches.

The Bricks work [5] made dynamic association between digital properties and physical handles through the tray device. Also an inspiration for the mediaBlocks work, the Bricks work did not develop physical objects as digital containers or physical manipulation outside of the "Active Desk" context.

Bishop's Marble Answering Machine [3] made compelling demonstration of passive marbles as "containers" for voice messages. Later work by Bishop demonstrated physical objects associated with diverse digital content, and prototyped an early object-GUI gateway.

The Pick-and-Drop work of [11] provides strong support for file exchange between palmtop, desktop, and wall-based GUIs with a pen stylus. However, the technique less directly addresses media exchange between non-GUI devices, or with devices that are not spatially adjacent.

Molenbach's LegoWall prototype (discussed in [6]) used LEGO structures to contain information about ocean-going ships. These objects were combined with display and control objects that, when plugged adjacent to containers, could display shipping schedules, send this data to hardcopy printers, etc.

The AlgoBlock system uses the manipulation of physical blocks to create computer programs [13]. Connections between the blocks create LOGO programs that may be rearranged by the user, esp. in an educational context.

The Digital Manipulatives research of Resnick, Borovoy, Kramer, et al. [12] has developed "societies of objects" including badges, buckets, beads, balls, and stacks. Each is associated

with digital semantics responsive to physical manipulations such as shaking, tossing, and stacking objects, or dunking objects in buckets of "digital paint." The manipulatives work makes strong progress towards developing objects with rich digital/physical couplings.

4 SYSTEM OVERVIEW

Figure 6 illustrates our current implementation of the mediaBlocks interface. The center column, media devices, lists the physical devices for which we have integrated mediaBlock support. The left column shows devices supporting operation of mediaBlock slots. The right column presents the computers that manage media recording and playback. The media browser and sequencer SGI machines play additional roles as the controllers for these tangible interfaces. However, these details are beyond the scope of our illustration.

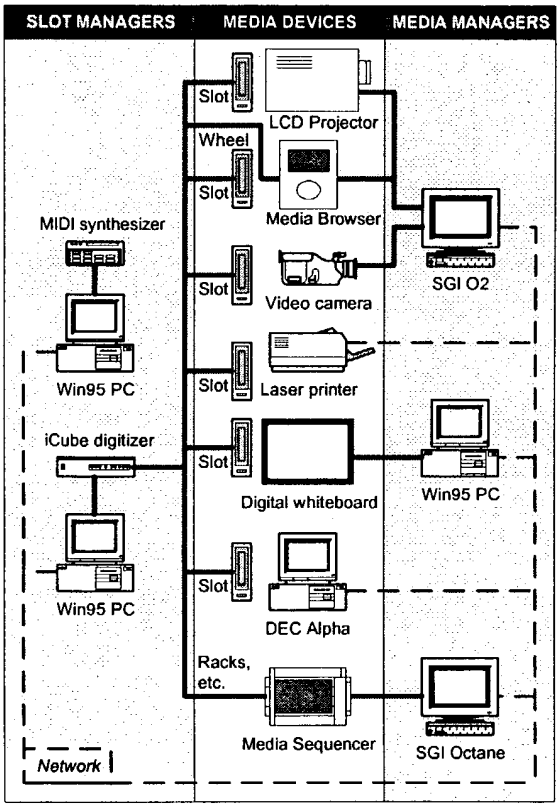

Figure 6: First-generation mediaBlocks system diagram

The mediaBlocks system was implemented with a Tcl- and [incr Tcl]-based tangible interface software/hardware toolkit called 3wish [14]. 3wish includes modules supporting MIDI-based digitizers and synthesizers, Inventor-based 3D graphics, computer vision and magnetic field trackers, etc. 3wish also supports a distributed architecture called *proxy-distributed* or *proxdist computation* [15], which provides strong abstractions for mixed physical/digital systems such as the mediaBlocks interface.

Development of 3wish support for the media sequencer and mediaBlock slots has consumed a large part of the mediaBlocks effort. The mediaBlock idea of objects as physical proxies for online content and computation grew out of early proxdist research, and the influence of 3wish's distributed architecture is visible in the half-dozen computers and tangible interfaces composing Figure 6. However, detailed discussion is beyond the scope of this paper, and is left to [14,15] and future treatments.

5 PHYSICAL CONTAINERS

The paper has emphasized the role of mediaBlocks as "containers" for media, as well as the use of mediaBlock slots for media interchange between various devices. However, a range of removable media devices have realized this basic function. For instance, videotapes and floppy disks are both "physical containers" for electronic media that support media interchange through systems of "physical slots." How do mediaBlocks relate to these well-known technologies?

The comparison will be explored for floppy disks (and other removable media analogs), which share the ability to store digital media of various formats. First, it is clear that mediaBlocks and floppy disks are technically quite different. Floppy disks function by taking information offline, recording media onto the disk's own storage. MediaBlocks instead work by moving information online, referenced by the internal ID of the mediaBlock object.

It is also interesting to note that mediaBlocks transparently support media with widely varying bandwidths. For instance, mediaBlocks are equally practical for digital whiteboard and digital video recordings, even though the characteristic bit rates differ by five orders of magnitude (~100KB vs. ~10GB per hour).

From a user interface standpoint, mediaBlocks and floppy disks also have a number of differences. Floppy disk contents are accessed indirectly through graphical or textual interaction on a host computer. In contrast, mediaBlock contents may be accessed through physical manipulation of the mediaBlock object itself. For example, inserting a target mediaBlock into a digital whiteboard's slot initiates recording "into" the block. Similarly, moving a host mediaBlock on our media sequencer's position rack allows sequences of images to be navigated (see section 6.1).

Building from this distinction, mediaBlocks possess a simplicity and "lightweight" mode of operation rarely found with the floppy disk medium. Media recording, playback, and exchange between our example whiteboard, camera, printer, projector, and computer are all as simple as inserting and withdrawing a mediaBlock from the respective slots.

The mediaBlock support for physical media exchange does not force a "sneaker-net" ethic upon users. Instead, mediaBlocks offer the simplicity and directness of physically referencing *which data* and *which device* when physical proximity is a convenient delimiter. Common "reference in absence" tasks such as dispatching jobs to remote printers for later pick-up or delivery may be supported by shortcut controls (e.g., a "print" button on the whiteboard), or by inserting mediaBlocks into TUI or GUI devices providing remote printer access.

Thus, mediaBlocks act not as a *medium of storage*, but rather a *mechanism of physical reference and exchange*. In this sense,

the use of mediaBlocks more closely resembles the interactive process of "copy and paste" propagated out into the physical world than the storage medium of floppy disks. Conceptually consistent with the premise of tangible user interface, this is a major distinction that colors the spectrum of mediaBlocks applications.

Also in point of fact, neither floppy disks nor other removable media are native to our projector, printer, or whiteboard, nor do we know of these features in comparable products. The absence of media drives from these well established and commercially competitive products provides market evidence that mediaBlocks serve new or different conceptual roles.

5.1 Implementation

MediaBlocks are constructed as 5x5x2cm wooden blocks embedded with electronic ID tags. First-generation blocks are tagged with resistor IDs. New efforts use Dallas Semiconductor iButton™ devices, which incorporate both digital serial numbers and nonvolatile memory. It is worth noting that the physical form of mediaBlocks is a product of their intended use, not of technical limitations. Both tag technologies are available as surface-mount devices a few millimeters in diameter, rendering block size technically near arbitrary.

Both resistor- and iButton-based mediaBlocks couple to slots, racks, and other TUI elements with a pair of electrical contacts. For resistor-based blocks, ID is determined with a voltage-divider circuit against a reference resistor, sampled by an Infusion Icube MIDI A/D converter. For the iButton implementation, an interface using Microchip's PIC 16F84 microcontroller was implemented, based upon the iRX 2.0 board [10].

MediaBlocks operate in conjunction with a number of different interface devices. The most common of these is the slot. Slots were prototyped in foamcore, with copper tape contacts. Audio feedback for slot entrance, exit, and status events is currently supported by an external MIDI synthesizer.

MediaBlock phicons are separated from their media "contents" by several levels of indirection. These mappings include:

a) physical block → network address
b) network address → mediaBlock data structure
c) data structure element → individual media contents

The individual steps of this mapping process are illustrated in Figure 7. First, insertion of an ID-tagged block **(1)** is detected electronically by the slot **(2)**. The ID value is registered by a computer hosting the slot's tag reader **(3)**, and transmitted as a block entrance event to the display device's media manager **(4)**. The slot and media managers could be hosted on the same machine. In our implementation, the libraries supporting tag readers and media displays were specific to PC and SGI platforms, respectively, requiring separate computers for **(3)** and **(4)**.

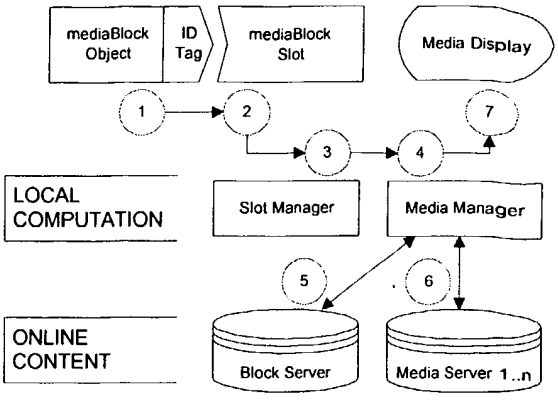

Figure 7: mediaBlock display flow diagram

Once an ID value has been obtained for a mediaBlock, the block server **(5)** provides a data structure describing the block's digital contents, presented in Table 1. With the resistor ID scheme, a central block server is responsible for mapping block IDs to block data structures for all compatible block devices. However, the resistor-based approach does not scale for distributed operation.

The iButton-based mediaBlocks solve this problem by storing the URL of their hosting block server within internal nonvolatile RAM (4096 bits for our chosen model), allowing truly distributed operation. iButtons also support storage of encrypted data, potentially useful for authenticating mediaBlocks and granting read or write permissions by a given media device.

After retrieving the block data structure, the device media manager retrieves the specified media contents from one or more media servers **(6)**. This content is finally sent to the media display under control of display-specific MIME registries **(7)**, similar to Web browser plug-in registries.

mediaList: *List of contained media element addresses*
physidType: *Type of physical ID tag on block*
physidInst: *ID of block tag (usually a number)*
mediaHost: *Media stored on media- or block-server?*
recordBehavior: *New media appends or overwrites old?*
lastObservedLocale: *Last location block observed*
lastObservedTime: *Timestamp of last block sighting*
blockLabel: *Text describing block contents*
blockLabelColor: *Color of paper block label*

Table 1: mediaBlock data structure

Earlier sections have discussed the use of mediaBlocks as a medium of exchange between graphical and tangible interfaces. This works particularly well in conjunction with Microsoft's Internet Explorer 4.0 "Internet shortcuts" feature (and equivalencies provided by other operating systems). "Internet shortcuts" allow distributed online media (e.g., URL-referenced HTTP sources) to be manipulated by the Windows95 desktop and applications indistinguishably from files on local disk drives.

We have explored the synthesis of Internet Shortcuts from media elements dragged out of monitor-slot mediaBlocks with GUI drag and drop. This combination represents a significant step towards truly seamless integration between the online media spaces of graphical and tangible interfaces.

6 PHYSICAL CONTROLS

The previous section discussed the physical containment, transport, and interchange aspects of mediaBlocks. The section also noted that in this capacity, the use of mediaBlocks more closely resembles the software process of "copy and paste" than the storage function of floppy disks.

However, the earlier section did not discuss how users might actively manipulate mediaBlock contents. For example, while both video decks and MS Windows95 CD-ROM drives support "auto-launch" capabilities analogous to mediaBlock slot playback, these systems also provide additional controls for more sophisticated interactions.

Following these examples, we might model interfaces on the physical play/stop buttons and jog/shuttle wheels of VCRs, as we have done with our media browser. We might also use traditional graphical interfaces to manipulate mediaBlock contents, as we have done with the GUI monitor slot.

At the same time, it is interesting to explore new interface possibilities specific to tangible user interfaces. The slot-based "copy and paste" functionality illustrates the first step of such an approach, binding digital semantics to the insertion of mediaBlocks into physical slots.

We have carried this approach forward in our work with the *media sequencer* interface. Here, we introduce *racks, stacks, chutes,* and *pads* as physical constraints that enable mediaBlocks to act as physical *controls* for directly manipulating block contents (Fig. 8).

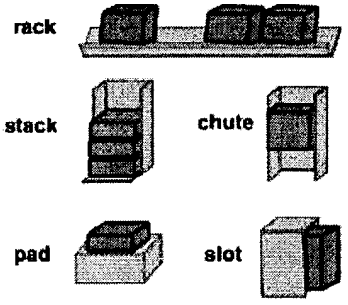

Figure 8: Media sequencer physical constraints,
used in combination with mediaBlocks as physical controls

The mediaBlock *rack* is the primary physical constraint elements used in the media sequencer. The mediaBlocks rack was inspired by the Scrabble™ board game's combination of letter tiles and the tile rack. In Scrabble, the rack allows letter tiles to be rapidly inserted, sequenced, and grouped into meaningful patterns. In the sequencer context, these physical attributes of position, sequence, and adjacency may be digitally recast as indexing, sequencing, and Boolean AND/OR operations, respectively.

6.1 User Interface

The media sequencer prototype is illustrated in Figures 5, 9, and 10. Sequencer operation is dominated by two physical constraint structures: the *position rack* and *sequence rack.*

When a mediaBlock is placed in the position rack, its contents are shown on the sequencer display as a perspective wall [9]. The focus position of the perspective wall is interactively controlled by the relative position of the mediaBlock on the position rack (Figure 10). Moving the block to the rack's left edge moves the perspective wall to focus on the block's first element. Similarly, the right edge of the position rack corresponds to the block's last element.

The combined position rack and perspective wall serve several purposes. First, they support the interactive viewing of mediaBlock contents with imaging of both detail and context [9]. Secondly, they allow the position rack to select an individual media element for copying between mediaBlocks.

Towards this end, a destination mediaBlock can be placed in the sequencer's *target pad,* physically adjacent the position rack (see Figure 9). Pressing the block on the target pad will append the currently selected media element into the destination block. This is confirmed with audio feedback and a haptic "click," as well as an animation of the selected media transferring "into" the target block. Pressing and holding the block, followed by moving the source mediaBlock to a new location, will copy a range of elements into the target block (analogous to dragging out a selection range with a mouse).

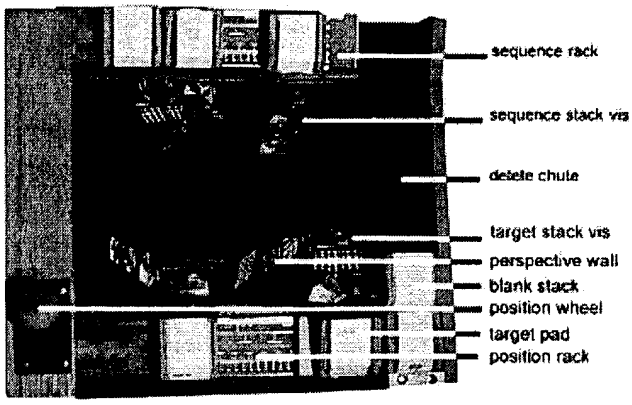

Figure 9: Media sequencer components

We believe this concept of using mediaBlock phicons to physically manipulate their internal contents generalizes to a powerful new interaction technique. Here, we are using mediaBlocks as *physical controls* for directly acting upon their internal state. The nearest equivalent in GUIs might be dragging a graphical folder icon across a scrollbar to index through folder contents. While a somewhat bizarre behavior for the GUI, we believe that use of mediaBlocks as physical controls holds substantial promise in the TUI context.

The *sequence rack* extends the control functionality of mediaBlocks. This rack allows the user to combine the contents of multiple mediaBlocks into a single sequence, which can then be associated with a new mediaBlock carrier on the target pad. When a mediaBlock is placed onto the sequence rack, its con-

tents scroll out of the block into the sequencer display space (Figure 9). Multiple mediaBlocks may be arranged to construct a new sequence.

Figure 10: Media sequencer perspective wall (alternate view) *associated with movement of mediaBlock on position rack; a compact disk's music is "contained" within this block*

Both the sequencer screen and target pad are shared between the position and sequence racks. When a mediaBlock is located on the position rack, the target pad is bound to selections from the perspective wall display. When the position rack is clear, the target pad is associated with recording of the sequence rack's aggregate elements.

The use of mediaBlocks as physical controls services only part of the sequence rack's behavior. Especially when a source mediaBlock contains many elements, navigating the perspective wall by incremental steps may be more convenient than using the position rack. The *position wheel* supports such incremental navigation. Haptic detents provide the user with physical feedback, where each detent corresponds with movement of one media element.

6.2 Implementation

The media sequencer platform was assembled of wood and acrylic, and embedded with a 1280x1024-pixel 32cm-diagonal flat panel display. Position and sequence rack sensing was performed with a custom contact-grid board. The position wheel was tracked with a shaft encoder. Sensor inputs were digitized with the Infusion Systems Icube device, and acquired through the 3wish extensions to the Tcl language [14] using C++ wrappers over the Rogus MIDI interface suite [4]. The position rack perspective wall and other graphical visualizations were written with 3wish's Open Inventor-based routines, and executed on an SGI Octane workstation.

7 DISCUSSION

While mediaBlocks make progress towards broader vocabularies for tangible interface design, the project also illustrates some of the challenges in designing TUIs. As a case in point, we consider some of the design decisions leading to our sequencer prototype's current form. Some readers have questioned the sequencer's integration of a central graphic display, given our research focus upon tangible interface.

While the sequencer indeed integrates a flat-panel display, we argue that the design is the product of a particular interface task and set of design constraints. Our motivating task was the sequencing of presentation media, especially images and video. Our interface inspirations were the Scrabble tile/rack constraint system, and the brick/tray function of [5]. As our task centered upon manipulation of visual media, a graphic display of some sort was essential.

Our original hope was to integrate this display into a rack's footprint, displaying the graphical contents of transparent mediaBlocks in a fashion following the metaDESK's passive lens [15]. However, given that mediaBlocks usually contain multiple media elements, we had difficulty determining an effective method of display within a block's 5x5cm footprint. Thus, we decided upon a visual display external to the mediaBlock/rack system, while maintaining visual contiguity with the associated mediaBlock container.

Here, we explored use of back-projected, front-projected, and integrated displays. While each had advantages, we selected a 32cm-diagonal 1280x1024-pixel integrated flat panel display on the basis of resolution, dot pitch, and compactness of integration. We wished adequate display resolution to support the visually-intensive task. We were also interested in a relatively small, compact device, even at the cost of greater display real estate such as provided by the metaDESK [15].

This was because we wished to make extensive use of physical constraints to support the sequencing task. Simultaneously, we imagined our users making simultaneous use of multiple complementary devices, such as the combination of a general-purpose computer for authoring new content, and the sequencer for assembling and manipulating the presentation. These usage constraints, coupled with the drive for proximity between physical controls and visual displays, drove the system to its current form.

Aspects of the sequencer design remain challenging, including potential inconsistency between the position wheel and rack; linkage between rack-based mediaBlocks and screen-based displays; and the shared screen use by position and sequence racks. Additionally, it is possible that the position rack + mediaBlock selection mechanism is less efficient than (say) directly selecting contents with one's finger. A second-generation sequencer design is under development to add this direct content-selection ability, increase display real estate, and rationalize the behavior of sequencer racks, while maintaining mediaBlock controls for sorting, sequencing, and transporting media into and out of the sequencer.

We believe mediaBlocks' underlying containment and transport functions are fundamentally sound. MediaBlocks' use as physical controls makes a significant extension to this transport function, and has been demonstrated useful in tasks like the video figure's presentation example. Thus, while less well-established than the transport function, our implementation leaves us optimistic about the interface potential of phicon controls.

8 FUTURE WORK

While part of our research motivation lies in seeking new paradigms for interface outside the well-explored GUI context, it is interesting to explore parallels between graphical and physical

interaction techniques. One such instance is the historical emergence of consistent interface behavior across multiple GUI applications, notably articulated in the Apple Human Interface Guidelines [1].

Our design of mediaBlock slots and sequencer constraints has been shaped by an interest in TUI analogs for GUI inter-application behaviors. Our use of mediaBlocks for media transport and interchange develops a physical analog of the GUI "copy-and-paste" functionality. Similarly, the sequencer's racks, chute, and other constraints have parallels to GUI "interaction primitives" (e.g., desktop-based clicking, dragging, etc. of icons), without directly embodying GUI widgetry as with the metaDESK [15].

As much of the TUI appeal lies in a diversity of physical embodiments to address specific interface tasks, it is unclear how far analogies to [1] might extend. Nonetheless, prospects for consistent TUI interface vocabularies and widespread TUI/GUI interoperability are highly attractive.

Finally, mediaBlock's online aspect suggests their ability to "contain" live, streaming content: in other words, the ability to operate as media "conduits." We have demonstrated mediaBlocks containing both streaming media sources such as RealAudio™ and RealVideo™ media, as well as pairs of mediaBlock conduits which together broadcast and receive streaming video.

At the same time, the conduit functionality of mediaBlocks represents a significant conceptual expansion with its own user interface questions. For instance, if a user wishes to both record and broadcast a whiteboard session, or both display and store a live video stream, how are these aggregate behaviors best accommodated? These and other open questions remain. As a result, we leave conduits for further discussion in future work.

9 CONCLUSION

We have presented a system of tangible user interface based upon *mediaBlocks*: small, electronically tagged wooden blocks that serve as physical containers, transports, and controls for online media. MediaBlocks do not store media internally, instead serving as phicons (physical icons) which give physical embodiment to online media.

MediaBlocks provide a mechanism for seamlessly exchanging digital contents between diverse media devices, including media sources, displays, general-purpose computers, and specialized tangible interfaces. Towards these ends, we have demonstrated the ability of mediaBlocks to physically "copy" media from a whiteboard and camera source, and "paste" this content into a printer and projector.

We have also shown the use of mediaBlock slots as physical gateways between tangible and graphical interfaces, allowing media to be swiftly exchanged with traditional GUIs.

Additionally, mediaBlocks are used as physical controls for operating upon their own digital "contents." We have demonstrated this ability in the media sequencer by swiftly composing a presentation integrating video, photographs, whiteboard recordings, and text. While the sequencer supports this task with a graphical display, the entire task is accomplished without a keyboard, pointer, or cursor, except for the GUI-based entry of the textual slide.

Finally, we have discussed analogs between TUI media exchange between diverse physical devices, and GUI support for consistent multi-application operation and communication. Similarly, we have demonstrated new roles and visions for seamless interaction with online content beyond the traditional GUI context.

In conclusion, we believe mediaBlocks are a powerful tangible interface for the seamless exchange and manipulation of online content between diverse media devices and people. More generally, we believe mediaBlocks represent a step towards broader tangible interface use as an interaction technique uniting people, computational media, and the physical environment.

10 ACKNOWLEDGEMENTS

Many people have contributed to this work. John Alex and Paul Grayson helped with software and hardware implementation. John Underkoffler and Paul Yarin assisted video production. Andrew Dahley and Andrew Hsu provided design assistance. Matt Gorbet, Scott Brave, and Victor Su provided additional helpful input. Ben Denckla helped with early Rogus efforts. Paul Grayson, Hannes Vilhjalmsson, Joe Marks, Tara Rosenberger, and others provided feedback on paper drafts. This research was sponsored in part by the MIT Media Lab's Things That Think consortium and a Mitsubishi fellowship.

11 REFERENCES

[1] Apple Computer, Inc. *Apple Human Interface Guidelines: The Apple Desktop Interface.* New York: Addison Wesley, 1987.

[2] Brave, S., and Dahley, A. inTouch: A Medium for Haptic Interpersonal Communication. In *CHI'97 Extended Abstracts,* pp. 363-364.

[3] Crampton Smith, G. The Hand That Rocks the Cradle. *I.D.*, May/June 1995, pp. 60-65.

[4] Denckla, B. Rogus C++ MIDI Suite. `http://theremin.media.mit.edu/rogus/`

[5] Fitzmaurice, G., Ishii, H., and Buxton, W. (1995). Bricks: Laying the Foundations for Graspable User Interfaces. *Proc. of CHI'95,* pp. 442-449.

[6] Fitzmaurice, G. *Graspable User Interfaces.* Ph.D. Thesis, University of Toronto, 1996.

[7] Gorbet, M., Orth, M., and Ishii, H. Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography. In *Proc. of CHI'98.*

[8] Ishii, H., and Ullmer, B. Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms. In *Proc. of CHI'97,* pp. 234-241.

[9] Mackinlay, J., Robertson, G., & Card, S. The Perspective Wall: Detail and context smoothly integrated. In *Proc. of CHI'91*, pp. 173-179.

[10] Poor, R. The iRX 2.0 ...where Atoms meet Bits. `http://ttt.media.mit.edu/pia/Research/iRX2/`

[11] Rekimoto, J. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proc. of UIST'97,* pp. 31-39.

[12] Resnick, M., Berg, F., et al. Digital Manipulatives: New Toys to Think With. In *Proc. of CHI'98.*

[13] Suzuki, H., Kato, H. AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning. In *Proc. of 4th European Logo Conference,* Aug. 1993, Athens Greece, pp. 297-303.

[14] Ullmer, B. 3wish: Distributed [incr Tcl] Extensions for Physical-World Interfaces. In *Proc. of Tcl/Tk'97,* pp. 169-170.

[15] Ullmer, B., and Ishii, H. The metaDESK: Models and Prototypes for Tangible User Interfaces. In *Proc. of UIST'97,* pp. 223-232.

[16] Weiser, M. The Computer for the 21st Century. In *Scientific American,* 265(3), pp. 94-104.

Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography

Matthew G. Gorbet
Tangible Media Group
MIT Media Laboratory
20 Ames Street
Cambridge, MA 02139 USA
+1 617 253 0128
mgorbet@media.mit.edu

Maggie Orth
Hyperinstruments Group
MIT Media Laboratory
20 Ames Street
Cambridge, MA 02139 USA
+1 617 253 0804
morth@media.mit.edu

Hiroshi Ishii
Tangible Media Group
MIT Media Laboratory
20 Ames Street
Cambridge, MA 02139 USA
+1 617 253 7514
ishii@media.mit.edu

ABSTRACT

This paper presents a system for interacting with digital information, called Triangles. The Triangles system is a physical/digital construction kit, which allows users to use two hands to grasp and manipulate complex digital information. The kit consists of a set of identical flat, plastic triangles, each with a microprocessor inside and magnetic edge connectors. The connectors enable the Triangles to be physically connected to each other and provide tactile feedback of these connections. The connectors also pass electricity, allowing the Triangles to communicate digital information to each other and to a desktop computer. When the pieces contact one another, specific connection information is sent back to a computer that keeps track of the configuration of the system. Specific two and three-dimensional configurations of the pieces can trigger application events.

The Triangles system provides a physical embodiment of digital information topography. The individual tiles have a simple geometric form which does not inherit the semantics of everyday physical objects. Their shape, size, and connectors encourage rapid rearrangement and exploration of groups of Triangles. The infinitely reconfigurable 2D and 3D topographies of the Triangles system create a new language for tangible interface.

Keywords

Interface design, tangible interface, physical interface, graspable interface, digital connector, physical connector, magnetic connector, tangible bits.

INTRODUCTION

Physical building blocks are powerful tools for thought and play. They allow quick construction and manipulation of structures through two-handed tactile interaction. This enables an easy exploration and understanding of the blocks' interactions with one another, and also of physical properties like balance, gravity, composition, and form (Fig. 1).

CHI 98 Los Angeles CA USA
Copyright 1998 0-89791-975-0/98/4..$5.00

However, these blocks are usually passive objects, and computers can not recognize or interpret the composite structures of blocks.

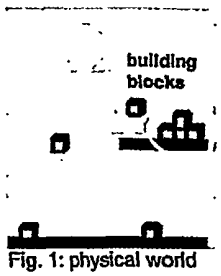

Fig. 1: physical world

In comparison, personal computers allow users to create, edit, and store complicated digital information structures. Manipulating these digital objects is heavily graphical-display-oriented, with the information relationships represented visually on a screen. Although computers support malleable information manipulation, current interactions are largely constrained by the mouse and keyboard (Fig. 2). Shifting focus between the 'controls' of the computer (keyboard and mouse) and the 'data' (as represented by visual images on the screen) can be a source of confusion and delay in working with digital information [7]. The affordances of physical building blocks are inherently richer than what current GUI affords through conventional "direct manipulation" techniques. How can we get the best of both the digital and physical worlds?

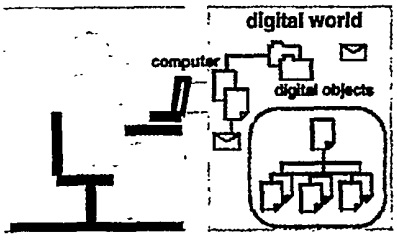

Fig. 2: looking into the digital world

Increasingly, we have seen an interest in physical systems for interacting with information [2, 3, 6, 11, 12, 13]. The Triangles system was designed to address this issue by physically embodying the *topography* of digital information – the relationships and connections between data elements – so that users can directly feel and manipulate the digital information with their two hands (Fig. 3). In the same way that building blocks help explore physical properties like balance and composition, Triangles were designed to provide easier interaction with and understanding of data space.

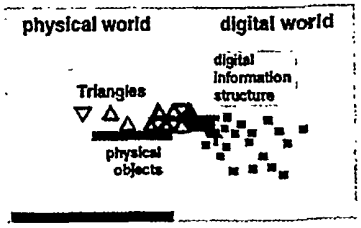

Fig. 3: manipulating the digital world

Unique contributions of the Triangles system are the following:

- The introduction of a concept of joint digital/physical objects which, by design, embody information and provide a general physical means to make a computer aware of their topography
- A user interface that
  - takes advantage of the human ability to easily manipulate physical objects with both hands,
  - provides a persistent physical indicator of state, and
  - allows many people to use it simultaneously.
- A new design for an electrical and magnetic connection mechanism that provides
  - a low-bandwidth, computationally inexpensive means for sensing the manipulation of complex physical structures, and
  - tactile feedback to users in attach and detach operations.

In this paper, we introduce the Triangles system, along with four Triangles applications. We discuss salient features of these applications and lessons learned from developing them. We also describe the iterative design and development of the Triangles system itself, including hardware, software and conceptual design. We begin with a brief scenario from one of our current Triangles applications.

CINDERELLA

Two children are sitting on the floor, with a set of ten triangular tiles in between them. On the tiles are images from the popular fairy-tale Cinderella. One child spreads the tiles out on the floor, looking through them. She picks one up, with an image of Cinderella's evil stepmother on it. The other child hands her a tile with Cinderella's house on it, and she brings them together. They snap together magnetically with an audible click, and instantly the step-mother's voice is heard (emanating from a set of nearby speakers) echoing through the halls of the house: "Cinderella! Cinderella?! Oh, where is that girl?..."

The stepmother's voice continues, calling out for Cinderella, until one of the children picks up the image of Cinderella sweeping, and snaps it onto another edge of the stepmother triangle.

"Ah, there you are! I need you to clean up this messy room. And when you're done with that, there's laundry to fold downstairs," says the stepmother. Cinderella obeys, and sounds of sweeping and humming are heard. Next, the children remove the stepmother. As soon as she is gone, Cinderella explains how she longs to be free from her stepmother's tyrannical rule. Just then, the children attach an image of the stepsisters, and we hear them burst into the room, teasing Cinderella...

This scenario is an example of an interaction with one of the storytelling applications that was developed for the Triangles system. These applications will be discussed in detail later in this paper.

THE TRIANGLES SYSTEM

Overview

The Triangles system consists of a set of identical flat, plastic equilateral triangles, each with a microprocessor inside and a unique digital ID. The Triangles each have different images or markings on them, which can be changed depending on the intended application. They have magnetic connectors on their edges which allow easy physical interconnection, as shown in Figure 4. The connectors also pass electricity, and the Triangles use them to communicate digital information to each other and to a desktop computer. Thus, when the pieces contact one another, specific information about that connection is sent back to the computer, which keeps track of the history of connections and current topography of the system.

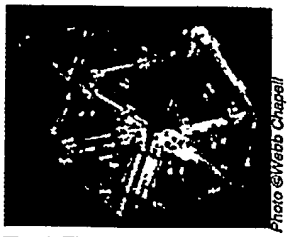

Fig. 4: The Triangles

The triangles can be used to make two- and three-dimensional objects whose exact configuration is known to the computer. Changes in this configuration can trigger specific computer-controlled events. For example, connecting or disconnecting two specific Triangles could provide access to a specific web page, or cause a digitized audio sample to play. Events can also be associated with specific groupings of Triangles, rather than simple connection or disconnection of individual triangles. The actual output event that results from a given interaction depends on the application being used, but can be practically anything that a computer can control. The roles played by each part of the Triangles system are illustrated in Figure 5.

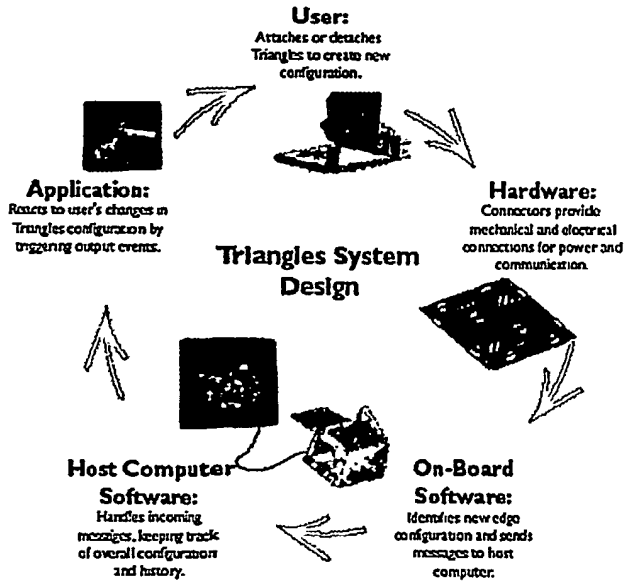

Fig. 5: Elements of the Triangles system and their roles.

The triangular tiles that make up the Triangles system have many affordances that help them to create a new language for interacting with digital information:

- They are easily handled and sorted with two hands.
- They can tile or make branching structures in two dimensions or create three-dimensional polyhedra.
- They can be manipulated by more than one person at a time.
- Their connections remain physically persistent, as a physical reminder of the state of the system.
- They provide tactile feedback when connections are made or broken, through their magnetic connectors.
- Their very general shape allows them to represent any type of digital information without carrying the semantic weight often associated with everyday physical objects.

Why Triangles?

The aim is to create a generic object that can represent the topographical relationships of information elements. Keeping the form of the objects generic enables them to be all about connections and relationships – a simple geometric shape has less semantic loading [1] than familiar objects such as a digitally augmented set of dolls or books. Pieces that can be tiled are appropriate to physically embody the idea of connection or association, and the flat faces of tiles enable them to carry a pictorial representation of the information which they represent.

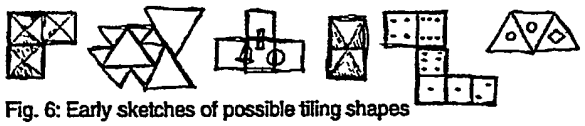

Fig. 6: Early sketches of possible tiling shapes

With these concerns in mind, many shapes were considered for the individual tiles (Fig. 6). Equilateral triangles are the simplest shape that can tile in two dimensions, so triangular tiles have the minimum number of sides required to physically reflect the complex possibilities of branching structures and relations in computer information. The three sides of a Triangle suggest junctions with one input and two outputs, or vice-versa. In contrast, objects with only two points of connection, like stacking blocks, suggest structurally linear data relationships, with only one input and one output. Another advantage to Triangles over other shapes like squares is that the triangular pieces allow robust three-dimensional (polyhedral) forms to be built.

Even though the interaction with Triangles is fairly simple, allowing users to quickly create and manipulate associations between the elements, the configurations that can be created are quite complex, due to the combinatorics of equilateral triangles. Since any edge of each tile can create a uniquely identifiable physical connection with any edge of another triangle, the number of unique configuration possibilities for a set of tiles is given by:

$$x \geq \frac{3^n(n+1)!}{6}$$

where n is the number of triangles and x is the number of possible unique configurations.

This means that for a set of just four triangles there are 1,620 possible 2D configurations. As the number of triangles increases, this number of configurations quickly grows to millions. Such a potentially vast world of information combined with the simplicity of interaction on the part of the user makes Triangles flexible and powerful, while creating interesting design issues for applications.

APPLICATIONS

The Triangles system presents a very specific means for interaction, lending itself to certain types of applications over others. For example, while it might be very well suited to exploring a non-linear narrative or configuring an audio/visual system, it would not be appropriate for applications which require a rich input vocabulary, like text editing or technical illustration.

The Triangles system has been seen and used by hundreds of people during the first year of its development. These include researchers from a wide variety of fields, artists, children visiting our laboratory, and representatives of many diverse industries. Our observations of these users' interactions with the system have contributed a great deal to its development. Following, we discuss four applications that we developed for the Triangles system, and what we learned from each.

Non-Linear Storytelling

The first applications that were developed for the Triangles system (March/April 1997) were storytelling systems.

Galapagos! – A World-Wide Web Story

In *Galapagos!*, partial illustrations of characters, places and events are placed on the faces of the Triangles in such a way that one or more users connecting two edges together can complete these images. As the two halves of a character or event are connected, web pages containing the content of the story appear on the user's screen. Which triangles are chosen and the order in which they are connected to one another determine aspects of the progression of the story. The result is a non-linear narrative that is told partially by a comic book-like arrangement of physical tiles, and partially by animated images and text on a computer screen (Fig. 7).

One problem with *Galapagos!* is that its content is entirely visual, requiring the user to split their visual focus between the images and text on the computer screen and the Triangles themselves. Children who played with the application did not always know where they should be looking, and expected audio feedback. This issue was addressed in the next storytelling application that we created, *Cinderella 2000.*

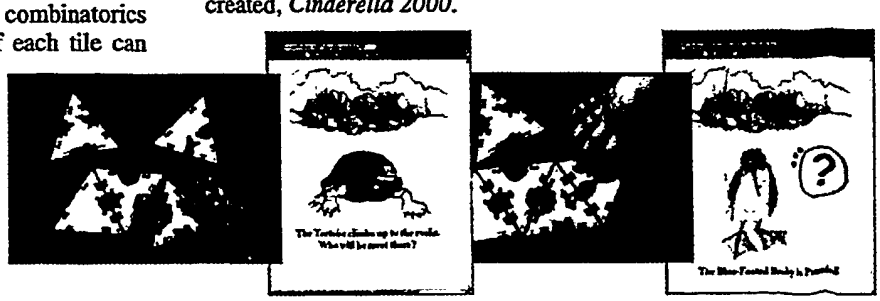

Fig. 7: Interacting with *Galapagos!*, a user connects Triangles to access web content.

Cinderella 2000 – An Audio Comic Book

Cinderella 2000 presents a modern version of the Cinderella fairy tale. Interactively arranging seven triangles that depict various aspects of the story, user can trigger audio samples stored on a desktop computer, creating a soundtrack of sound effects, narration and dialogue in the voices of the characters. These sounds are synchronized with the progression of the story, because they are triggered by specific connection events and Triangle configurations. Using audio for the output avoided the split-focus experienced with *Galapagos!*, creating a more compelling and complete storytelling experience. One interaction scenario for Cinderella 2000 is described at the beginning of this paper.

The images for *Cinderella 2000* were more varied in their arrangement and design than those in *Galapagos!* The design of the visuals was greatly influenced by the techniques and visual language of comics [9], making use of framing, scale, implied action and composition to create a narrative progression through still frames (Fig 8).

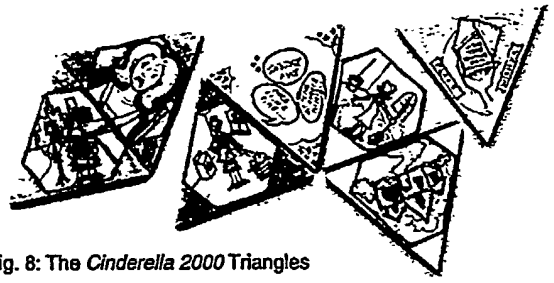

Fig. 8: The *Cinderella 2000* Triangles

Also, two specific Triangles were created as interaction devices:

- *Event Triangles*, symbolizing specific events in the story, for example the arrival of the invitation to the ball. Attaching an event triangle changes the context of the story, and thus the behavior of the characters.
- An *Info Triangle*, depicting three comic-book 'voice bubbles'. Attaching a specific edge of this triangle to a character would cause that character to reveal certain information about themselves.

Lessons Learned from Storytelling Applications

With *Galapagos!* and *Cinderella 2000*, we showed the potential of the Triangles system as a general interface for non-linear storytelling. In creating these applications, we also explored techniques and developed general authoring tools that would be useful for others creating nonlinear Triangles content.

However, one thing that became clear from implementing these two applications was the difficulty inherent in authoring unique content for the astonishing number of configurations possible with the Triangles system. The *Galapagos!* and *Cinderella 2000* applications each used seven triangles, offering literally millions of possible unique configurations. It was clear that limiting the number of 'appropriate' connections was necessary, in order to avoid having to create a huge number of unique content events. This was addressed in *Galapagos!* by using the partial illustrations to suggest which connections would be appropriate. Still, interaction issues arose around what would happen if 'incorrect' connections were made. For example, connecting half of a turtle to half of a bird might seem reasonable in a fantasy story about mythical animals.

If the application could respond appropriately to any of hundreds of thousands of possible connections, some extremely compelling and interesting storytelling applications might be possible. Advanced artificial intelligence and emergent behavior research [8] suggests that such applications could actually be written, generating or modifying content on the fly and thus making full use of the Triangles system's potential. In the future, we hope to collaborate with experts in this field to further investigate this possibility.

Another critical lesson learned from the storytelling applications was the importance of providing a single focus for the user's attention. The use of audio feedback was much more effective with children than pure visual content, as discussed above.

Triangles provide a very simple means for interacting with a potentially very complicated set of character relationships and storytelling situations. This ease of manipulation can also be applied to other sorts of information relationships. The next applications we developed use Triangles as an interface for configuring complicated media systems.

Media Configuration

TriMediaManager

Triangles' potential as a control system for information was further explored in *TriMediaManager*, an application in which the Triangles system is used to select and configure various media during a broadcast lecture or presentation. Triangles are given markings representing 'content' – audio and video clips, 3D datasets, images and other documents to be used during the session. The triangle that is directly attached to the computer is labeled as the *display*. During the session, the presenter can interactively decide which media is being shown by physically rearranging the positions of the triangles in relation to each other and to the *display* Triangle.

For example, if a presenter has access to a live video feed of herself, an audio clip, two video clips and a variety of images (presentation slides, for example), these can each be selected at any time by simply attaching the appropriate tile to the *display* triangle. If the presenter wishes to present several of these media in parallel, this can be achieved by joining the content triangles with one another. When this occurs, *TriMediaManager* attempts to simultaneously display as much of the total content as possible, giving precedence according to the proximity of each tile to the *display* triangle. In our example, if the tile representing the live feed of the presenter were directly connected to the *display* triangle, it would be broadcast as a full-screen image with audio. If an *image* tile were then connected to another edge of the *live video* triangle, it would appear as a smaller image, inset into the video feed (Fig. 9). Connecting a *video clip* triangle on the remaining edge of the *live video* tile would inset the appropriate video clip in another corner of the screen (the audio streams from the live feed and the clip would be mixed).

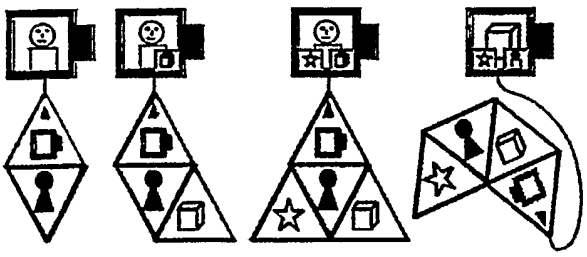

Fig. 9: *TriMediaManager*

At any point in the presentation, the presenter could easily change the display priorities of the various media: Moving the *display* triangle so that it was connected directly to the *video clip* tile would cause the hierarchy to shift, and the output to respond accordingly: the clip would take over the full screen, with the image and the live video feed each inset as smaller windows.

The presenter might attach still more triangles to the configuration, which might or might not be immediately displayed, depending on available output resources and proximity to the *display* triangle. The *TriMediaManager* application keeps track of what display resources are available (inset windows, audio channels, volume control, etc) and how each type of information is being displayed (i.e. moving video, still image, audio, or rendered object) and allocates resources for as much of the data as possible.

The human ability to intuitively manipulate, sort and arrange physical objects is exploited in *TriMediaManager*, creating a physical interface for accessing and arranging the display of a great deal of presentation information in realtime. Using the Triangles to represent high-level content selection like 'vacation clip' or 'earnings slide' as opposed to using traditional patching controls like 'VCR1→MON2' or 'carousel advance' allows the presenter to focus on content and dynamically rearrange their presentation if the need arises.

Artistic Expression

In creating *Galapagos!*, *Cinderella 2000*, and *TriMediaManager*, we explored many of the benefits of the Triangles system, including the exploratory nature of rearranging Triangles, the combinatorial potential of Triangles configurations, and the narrative potential of spatial tile arrangements.

One drawback common to all of these applications is their use of pre-defined mappings of information to Triangles. This requires extensive content authoring before each application can be used. *The Digital Veil*, the next Triangles project that was undertaken, allows users to control not only the output generated by specific Triangles interactions, but to assign and reassign meaning to groupings of Triangles during the course of an interaction. *The Digital Veil* was created as an art installation for the 1997 *Ars Electronica* festival in Linz, Austria.[4]

The piece consists of a table on which are laid out 35 Triangles. Each tile has a photograph, illustration, graphic symbol or physical texture applied to its surface(Fig. 10). These elements were designed to be beautiful, evocative and meaningful, both individually and in combination with each other. The public is invited to interact with the tiles in two locations on the table (Fig. 11), creating arrangements of Triangles and connecting them to the 'input station' or the 'output station' – two small boxes on the table with exposed triangle-edge connectors. The 'input station' has a light-up button and a microphone on it. When a user connects up to four triangles together and attaches this arrangement to the input station, the button lights up, and the user can push it and speak into the microphone. Their voice is sampled and linked with the specific arrangement of Triangles that they created. In this way, participants can 'assign meaning' to their configurations, creating illustrated phrases and small narratives that hold personal meaning.

Photo: rubra

Fig. 10: Images used in *The Digital Veil*, designed to be beautiful and evocative.

At the 'output station,' participants can create large configurations of triangles, building a visual and tactile texture on the table in front of them. As they do so, each individual triangle that they add triggers its own evocative audio sample, building an aural texture to accompany the configuration that they create. In creating the large configuration, if the user arranges any of the tiles to form one of the 'phrases' that had been recorded by a previous participant, that audio recording is also played back. In this way, the piece grows and changes over the course of its presentation, keeping a memory of the meanings and associations that users have created.

Fig. 11: Members of the Public Interacting with *The Digital Veil*.

The Digital Veil was created to address the reconfigurability of meaning that is inherent in digital information systems. Its use of Triangles allows participants to explore this in an interactive and creative way without needing to learn the particulars of a new user interface. The Triangles provide the feeling that users are actually holding and rearranging the information itself.

Audience members greatly enjoyed being able to quickly make new groupings of Triangles that they could personalize with their voice. They found their own uses for the application, sometimes leaving 'secret messages' in the system to be retrieved later, or even singing 'rounds' that could be controlled by adding new Triangles at the right time.

DESIGN AND DEVELOPMENT OF THE SYSTEM

The development of the system brought together skills from a variety of fields to give the Triangles functionality in the physical and digital worlds. Software architecture, electrical engineering, mechanical and industrial design converged to make the Triangles balanced physical/digital objects. One example of this balance is the physical form of the tiles, which suggest association and rearrangement (key elements of data manipulation). Another is the use of the physical properties of magnets to ensure good mechanical and digital connections while reducing ambiguity and providing tactile feedback to the user.

The design process was an iterative undertaking by two researchers, each with extensive prior interaction-design experience. One was well-versed in the physical and mechanical design of interactive objects, and the other had experience in software systems and traditional (GUI) user interface design. Close collaboration and a good understanding of each other's field of expertise enabled us to find innovative solutions to design challenges at many levels. A tight design loop was established, with new hardware, software and conceptual design maturing simultaneously. Following is a discussion of key conceptual, mechanical and software design issues faced in the development of the Triangles system.

Entropy and Physical Delimiters

When designing physical interfaces, it is important to consider the number of degrees of freedom and possible scenarios that can occur. This is sometimes called the *entropy* of the system. It is impossible to anticipate all of the potential interactions that people can have with real-world objects; for example, a user might pick up the interface and wave it around in the air, juggle it, or otherwise use it in unpredictable ways. Some interface systems, such as speech recognition systems or vision systems, continuously track of all of the user's interactions. These systems must determine *action delimiters* to differentiate between continuous action events in order for those actions to be correctly interpreted. Although powerful in certain situations, this kind of continuous physical sensing can be complicated and computationally expensive [10].

In the design of the Triangles, we sought to address the issues of entropy and action delimiters through the physical design of the objects themselves. Limiting the types of interaction that are suggested by the form of the object reduces the entropy of the system. The equilateral triangle shapes suggest tiling and edge-to-edge connections as appropriate ways of manipulating the objects, and the magnetic edge connectors reinforce this with tactile feedback when appropriate connections are made. Action delimiters are provided by the physical connection mechanism itself, which reports only *significant* changes to the system (connections and disconnections), as discrete events. These physical design decisions helped create a low-bandwidth and computationally inexpensive means for accurately interpreting a user's interaction with the Triangles system.

Physical Connectors


Fig. 12: Connector design sketches

The physical connectors had to be designed so as to make it easy to link any Triangle edge with any other edge in a robust, hinging configuration. Disconnecting the Triangles also had to be as simple as pulling them apart from one another. In addition, the connectors needed to provide a consistent supply of power and uninterrupted data connections between the microprocessors on the tiles.

Several designs for the physical connectors were examined, including slotted edges, snaps, zippers and conductive Velcro® fasteners (Fig. 12). Since equilateral triangles are radially symmetrical, connections such as magnets, snaps or Velcro, which have 'male' and 'female' components (or polarity, in the case of magnets) needed to be arranged so that 'male' would always meet 'female' and vice-versa (Fig. 13). Depending on the design of the electronic circuit, issues of symmetry arose in ensuring that each transmit pin would meet a receive pin, and that shared pins, such as power and ground, would always find the correct mate when connected.


Fig. 13

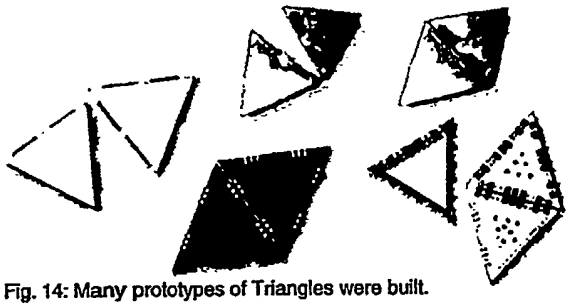
Fig. 14: Many prototypes of Triangles were built.

Many prototypes were built (Fig. 14), and eventually it was decided that the feel of the magnetic connectors was important to reinforcing the sensation of connection and disconnection. They make the triangles easy to attach and remove, and because they require no lateral motion, as do zippers or slots, connecting complex forms such as hexagons is easy. In addition, clever arrangement of the polarity of the magnets on the individual connectors can help force the edges into proper alignment with one another when making connections, and enables them to physically resist being wrongly connected. This avoids false connections and further reduces the entropy of the system.

Microprocessor Code

One of the Triangles in a given set acts as the *mother triangle*. It differs from the others in that it has a cable that provides power to itself and to the other triangles, as well as serial communication to a *host* computer. As soon as any Triangle is connected to this *mother triangle*, the new Triangle's microprocessor receives power through its edge connectors, and it can communicate with its neighbors and the host computer. The new Triangle then provides power and a communication pathway to subsequent Triangles as they are connected to it, and so on.

The microprocessor inside each Triangle is responsible for managing that Triangle's identity and information about its connections. When a Triangle receives power, its

microprocessor begins alternately polling and transmitting its ID to each of its edges. If any other Triangles are connected to it, they will be able to identify it and also the edge to which they are connected. Whenever such an event occurs, each Triangle involved generates a message containing the new configuration information, which it passes to the host computer. Disconnections between Triangles result in a similar message being passed by the Triangle that is still connected to the system.

The host computer receives these messages, and the Triangles system software layer interprets them. It reconciles connection messages that were generated simultaneously by two Triangles into *connection events* and keeps track of *disconnection events*. The system keeps a time-stamped history of all such events, and provides functionality, through an Application Programming Interface (API), for independent software applications to access and react to specific interactions with the Triangles.

Communication Structure

In order for the host computer (to which the Triangles are attached) to know the exact configuration of the system, a distributed network routing system was implemented. Each Triangle is responsible for determining when local events occur (connections and disconnections) and relaying information about these events back to the host computer, as discrete messages. The host computer then consolidates and reconciles these messages with one another in order to determine the configuration of the entire system.

To help develop and test the host software and various message-passing schemes in parallel with the development of the hardware, computer simulations (Fig. 15) of several routing algorithms were built, including:

- Probabilistic 'broadcasting' and regeneration of messages to every triangle with the expectation that the host would eventually receive the message;
- 'Maze-walking' which attempts to find the host connection by exploring all the possible connections;
- 'Topography maps' stored within each microprocessor to route the message at each step of its journey;
- 'Perimeter hugging' which attempts to find the edge of the configuration, and from there to trace a path back to the host;
- 'Gradient descent' which establishes gradients between each triangle and the host, and pass messages 'downhill' to ensure that they reach their goal.

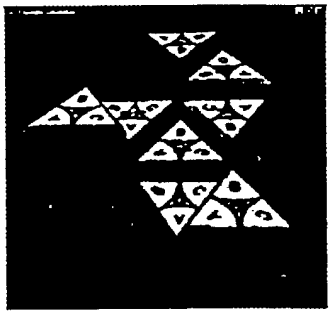

Fig. 15: Software Simulation of 'Gradient Descent'

Each of these algorithms had flaws, especially when implemented in a distributed system of low-memory microprocessors. For the initial implementation of the Triangles system, the 'gradient descent' algorithm [5] was used. This ensured that connection and disconnection messages would always be reported, but was unforgiving of errors caused by noisy connections, and introduced an undesirable amount of latency into the system.

Eventually, message-passing was simplified by making a hardware modification to the Triangles: A common serial bus was added to the architecture of the system, so that in addition to communicating with one another locally, any Triangle could pass messages directly to the host computer or to any other Triangle. This greatly improved the speed of the system and simplified the software design.

This illustrates a recurring theme in the design and development of the Triangles system. Almost every design challenge that was encountered could be solved in microprocessor software, electronics, mechanical hardware, or host software. Understanding the relationships between these elements and the trade-offs and balances between them was crucial to developing this system.

The Triangles API

A comprehensive API layer was developed for creating applications which use the Triangles system. The API layer consists of a C++ library of function calls and data structures that manage information about a user's interactions with the system.

The simplest way for developers to use the API is to specify certain Triangle *events* which the system should watch for, and provide functions in their application that should be called when these events occur. An *event* is either a connection or a disconnection between two Triangles. The event structure contains the following information:

- *Event type:* was this a connection or a disconnection?
- *Event time*: at what time did the event occur?
- *Triangle IDs:* what were the IDs of each triangle?
- *Triangle edges:* which edge of each triangle was involved in the event?

The event registration functionality is flexible, so that an application may specify some, all or none of these fields to monitor, allowing very general or very specific interactions to be specified. For example, an application might specify an action to be triggered when Triangle 4, side 2 is connected to Triangle 12, side 0, or simply any time Triangle 3 is connected or disconnected from anything else. (If none of the fields are specified, the application will be notified whenever *any* event occurs.) When a registered event does occur, the specified application function is called, and the details of the event that occurred are passed as parameters to the function. This enables the application to know exactly how the user interacted with the system.

The API also provides more comprehensive functionality. Using a simple command structure and a set of linked lists, an application can retrieve a complete description of which Triangles are present in the system and their connections. Once this list is provided, the application can query the API for information about any individual Triangles, including a complete history of that Triangles' interactions with its neighbors. In this way, an application can act not only on

events that are occurring in real-time, but also on past changes in the system. For example, if a Triangle is connected to another Triangle for the first time, it might trigger a different event than if the same Triangle is connected again later. If two Triangles are connected and another, specific Triangle is present in the system, this might have a different meaning than if the two Triangles were connected alone.

Using the API, developers need not be concerned with the message-passing protocols or internal functionality of the Triangles themselves. This creates a useful abstraction that allows Triangles applications to be developed by competent programmers from a variety of backgrounds. One of the aims of the Triangles project is to allow researchers in other fields such as computer science, design, education or physics to experiment with this tangible interface system.

FUTURE WORK

The Triangles system was conceived of as a foundation for exploring tangible interface in a broad variety of fields. Future work on this project will focus on providing the complete Triangles system to other researchers who are interested in developing interactive applications using the tangible interfaces.

Research areas that we have targeted at present include:

- Nonlinear narrative

  Further development of the storytelling potential of 'comic-book' Triangles with audio feedback.

- Groupwork Coordination

  Exploration of scheduling, group dynamics modeling and workflow systems using Triangles.

- Emergent behavior/AI models for content generation

  Using semantic webs and intelligent agent technology to interpret the meanings of various Triangles associations.

Continued refinement of the API is planned, along with the development of GUI and tangible authoring tools to support Triangles application development. Software ports of the API from its current Windows® platform to Silicon Graphics™, MacOS™, and general UNIX platforms will enable a broader variety of applications to be accessed with the Triangles.

CONCLUSIONS

We have presented a new tangible interface for the manipulation and exploration of digital information topography. Bridging the gap between the physical and digital worlds is not trivial. Rather than attempting to give computers a complete understanding of human language (voice, gesture) and the real world, our approach provides a new language which both people and computers can understand. Synchronization of what users have in their hands with the digital connections that computers can perceive occurs when the Triangle pieces are attached and detached from each other. Although this language (connect/disconnect) is quite simple, the power of expression enabled by a topography of multiple triangles is tremendous and very rich.

The concept of Triangles was born in October 1996. Many physical mockups of the Triangles and scenarios were created along with software models and simulations. This iterative development process led to the first working prototype in March 1997, which was demonstrated internally, with the sample application *Galapagos!* The benefits and limitations of Triangles became apparent through the development of several other applications and refinement of the system itself. Although we have not yet undertaken extensive usability testing with the Triangles system, our internal development has led us to be confident about this new language for tangible interface.

Acknowledgments

We would like to express our sincere thanks to the following people and organizations for their contributions to this project: Wendy Chien, Emily Cooper, James Hsiao, Chris Lopes, Betty Lou McClanahan, Gong Ke Shen, Paul Yarin, and the MIT Media Laboratory's Things That Think consortium.

REFERENCES

1. Csikszentmihaly, M. and Rochberg-Halton, E. (1981). "The Meaning of Things: Domestic Symbols and the Self", Cambridge University Press, 1981.
2. Feiner, S., MacIntyre, B. and Seligmann, D. (1993). Knowledge-based augmented reality. *Communications of the ACM.* 36, 7 (1993), 52-62.
3. Fitzmaurice, G. W., Ishii, H. and Buxton, W. (1995). Bricks: Laying the Foundations for Graspable User Interfaces. *Proceedings of Conference on Human Factors in Computing Systems (CHI '95)*, Denver, Colorado, ACM Press, 442-449.
4. Gorbet, M. and Orth, M. (1997). Triangles and The Digital Veil, *FleshFactor: Informationsmacschine mensch (1997 Ars Electronica Festival Catalog)*, Linz, Austria, SpringerWein-NewYork, 280-283.
5. Gorbet, M. and Orth, M. (1997). Triangles: Design of a Physical/Digital Construction Kit. *Proceedings of Designing Interactive Systems: Processes, Practices, Methods, and Techniques (ACM DIS '97)*, Amsterdam, ACM, 125-128.
6. Ishii, H. and Ullmer, B. (1997). Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. *Proceedings of Conference on Human Factors in Computing Systems (CHI '97)*, Atlanta, Georgia, ACM Press, 234-241.
7. Kurtenback, G., Fitzmaurice, G., Baudel, T., and Buxton, W. (1997). The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. *Proceedings of the ACM CHI '97 Conference on Human Factors in Computing Systems (CHI '97)*, Atlanta, Georgia, ACM Press, 35-42.
8. Maes, P. and Brooks, R. (1990). Learning to Coordinate Behaviors. *Proceedings of AAAI-90: The American Conference on Artificial Intelligence*, Boston, MA, AAAI Press/MIT Press, 796-802.
9. McCloud, Scott. "Understanding Comics." Kitchen Sink Press, 1994.
10. Pentland, A. P. (1996). Smart Rooms. *Scientific American.* 274, 4 (1996), 54-62.
11. Resnick, M. (1993). Behavior Construction Kits. *Communications of the ACM.* 36, 7 (1993), 64-71.
12. Suzuki, H., Kato, H. (1993). AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning. *Proceedings of 4th European Logo Conference*, Aug. 1993, Athens Greece, 297-303.
13. Wellner, P. (1993). Interacting with Paper on the DigitalDesk. *Communications of the ACM.* 36, 7 (1993), 87-96.

Bricks: Laying the Foundations for Graspable User Interfaces

George W. Fitzmaurice (1), Hiroshi Ishii (2) and *William Buxton (1, 3)*

(1) Dynamic Graphics Project
CSRI, University of Toronto
Toronto, Ontario,
Canada M5S 1A4
Tel: +1 (416) 978-6619
E-mail: gf@dgp.toronto.edu
E-mail: buxton@dgp.toronto.edu

(2) NTT Human Interface Lab
1-2356 Take,
Yokosuka-Shi, Kanagawa,
238-03 JAPAN
Tel. 81-468-59-3522
E-mail: ishii.chi@xerox.com

(3) Alias Research Inc.
110 Richmond Street East
Toronto, Ontario,
Canada M5C 1P1
Tel. +1 (416) 362-9181

Abstract

We introduce the concept of Graspable User Interfaces that allow direct control of electronic or virtual objects through physical handles for control. These physical artifacts, which we call "bricks," are essentially new input devices that can be tightly coupled or "attached" to virtual objects for manipulation or for expressing action (e.g., to set parameters or for initiating processes). Our bricks operate on top of a large horizontal display surface known as the "ActiveDesk." We present four stages in the development of Graspable UIs: (1) a series of exploratory studies on hand gestures and grasping; (2) interaction simulations using mock-ups and rapid prototyping tools; (3) a working prototype and sample application called GraspDraw; and (4) the initial integrating of the Graspable UI concepts into a commercial application. Finally, we conclude by presenting a design space for Bricks which lay the foundation for further exploring and developing Graspable User Interfaces.

Keywords:

input devices, graphical user interfaces, graspable user interfaces, haptic input, two-handed interaction, prototyping, computer augmented environments, ubiquitous computing

Introduction

We propose a new paradigm, Graspable User Interfaces, which argues for having some of the virtual user interface elements take on physical forms. Traditional graphical user interfaces (GUIs) define a set of graphical interface elements (e.g., windows, icons, menus) that reside in a purely electronic or virtual form. Generic haptic input devices (e.g., mouse and keyboard) are primarily used to manipulate these virtual interface elements.

The Graspable UIs allow direct control of electronic or virtual objects through physical artifacts which act as handles for control (see Figure 1). These physical artifacts are essentially new input devices which can be tightly coupled or "attached" to virtual objects for manipulation or for expressing action (e.g., to set parameters or to initiate a process). In essence, Graspable UIs are a blend of virtual and physical artifacts, each offering affordances in their respective instantiation. In many cases, we wish to offer a seamless blend between the physical and virtual worlds.

FIGURE 1. A graspable object.

The basic premise is that the affordances of the physical handles are inherently richer than what virtual handles afford through conventional direct manipulation techniques. These physical affordances, which we will discuss in more detail later, include facilitating two handed interactions, spatial caching, and parallel position and orientation control.

The Graspable UI design offers a concurrence between space-multiplexed input and output. Input devices can be classified as being *space-multiplexed* or *time-multiplexed.* With space-multiplexed input, each function to be controlled has a dedicated transducer, each occupying its own space. For example, an automobile has a brake, clutch, throttle, steering wheel, and gear shift which are distinct, dedicated transducers controlling a single specific task. In contrast, time-multiplexing input uses one device to control different functions at different points in time. For instance, the mouse uses time multiplexing as it controls functions as diverse as menu selection, navigation using the scroll widgets, pointing, and activating "buttons." Traditional GUIs have an inherent dissonance in that the display output is often space-multiplexed (icons or control widgets occupy their own space and must be made visible to use) while the input is time-multiplexed (i.e., most of our actions are channeled through a single device, a mouse, over time). Only one task, therefore, can be performed at a time, as they all use the same transducer. The resulting interaction techniques are often sequential in nature and mutually exclusive. Graspable UIs attempt to overcome this.

In general, the Graspable UI design philosophy has several advantages:

- It encourages two handed interactions [3, 7];
- shifts to more specialized, context sensitive input devices;
- allows for more parallel input specification by the user, thereby improving the expressiveness or the communication capacity with the computer;
- leverages off of our well developed, everyday skills of prehensile behaviors [8] for physical object manipulations;
- externalizes traditionally internal computer representations;
- facilitates interactions by making interface elements more "direct" and more "manipulable" by using physical artifacts;
- takes advantage of our keen spatial reasoning [2] skills;
- offers a space multiplex design with a one to one mapping between control and controller; and finally,
- affords multi-person, collaborative use.

BASIC CONCEPTS

Graspable UIs allow direct control of electronic objects through physical artifacts which we call *bricks*. The bricks, approximately the size of LEGO(TM) bricks, sit and operate on a large, horizontal computer display surface (the Active Desk, described later). A *graspable* object is an object composed of both a physical handle (i.e., one or more bricks attached) and a virtual object (see Figure 1).

The bricks act as specialized input devices and are tracked by the host computer. From the computer's perspective, the brick devices are tightly coupled to the host computer -- capable of constantly receiving brick related information (e.g., position, orientation and selection information) which can be relayed to application programs and the operating system. From the user's perspective, the bricks act as physical handles to electronic objects and offer a rich blend of physical and electronic affordances.

One Handle

In the simplest case, we can think of the bricks as handles similar to that of graphical handles in computer drawing programs such as MacDraw(TM) (see Figure 2a). A physical handle (i.e., a brick) can be attached to an object. Placing a brick on the display surface causes the virtual object beneath it to become attached (see Figure 2b). Raising the brick above the surface releases the virtual object. To move or rotate a virtual object, the user moves or rotates the attached brick (see Figure 3). Note that the virtual object's center of rotation is at the center of the brick.

FIGURE 2. (a) Traditional MacDraw-like application which uses electronic handles to indicate a selection. (b) Selecting using a brick.

FIGURE 3. Move and rotate virtual object by manipulating physical brick which acts as a handle.

A simple example application may be a floor planner (see Figure 5a). Each piece of furniture has a physical brick attached and the user can arrange the pieces, most likely in a rapid trial-and-error fashion. This design lends itself to two handed interaction and the forming of highly transient groupings by touching and moving multiple bricks at the same time.

Two Handles

More sophisticated interaction techniques can be developed if we allow more than one handle (or brick) to be attached to a virtual object. For example, to stretch an electronic square, two physical bricks can be placed on an object. One brick acts like an anchor while the second brick is moved (see Figure 4).

FIGURE 4. Two bricks can stretch the square. One brick acts like an anchor while the second brick is moved.

Placing more than one brick on an electronic object gives the user multiple control points to manipulate an object. For example, a spline-curve can have bricks placed on its control points (see Figure 5b). A more compelling example is using the position and orientation information of the bricks to deform the shape of an object. In Figure 6, the user starts off with a rectangle shaped object. By placing a brick at both ends and rotating them at the same time, the user specifies a bending transformation similar to what would happen in the real world if the object were made out of a malleable material such as clay. It is difficult to imagine how this action or transformation could be expressed easily using a mouse.

One key idea that the examples illustrates is that the bricks can offer a significantly rich vocabulary of expression for input devices. Compared to most pointing devices (e.g., the mouse) which only offers an x-y location, the bricks offer multiple x-y locations and orientation information at the same instances of time.

FIGURE 5. (a) Proposed simple floor planner application. (b) Many physical bricks are used for specifying multiple control points for creating a spline curve.

FIGURE 6. Moving and rotating both bricks at the same time causes the electronic object to be transformed.

RELATED RESEARCH

Some research and commercial systems have been developed with a similar graspable theme. In some sense, many of these emerging systems exhibit the property of ubiquitous computing [14] in which computation is embedded in many

physical artifacts and spread throughout our everyday environment. The following systems illustrate the push towards ubiquitous computing, physical manipulation interfaces and merging physical and virtual artifacts.

The LegoWall prototype (developed by A/S Modulex, Billund Denmark in conjunction with the Jutland Institute of Technology in 1988) consists of specially designed LEGO blocks that fasten to a wall mounted peg-board panel composed of a grid of connectors. The connectors supply power and a means of communication from the blocks to a central processing unit. This central processing unit runs an expert system to help track where the blocks are and what actions are valid.

The behavior construction kits [9] consist of computerized LEGO pieces with electronic sensors (such as light, temperature, pressure) which can be programmed by a computer (using LEGO/Logo) and assembled by users. These LEGO machines can be spread throughout the environment to capture or interact with behaviors of people, animals or other physical objects. The "programmable brick," a small battery powered computer containing a microprocessor, non-volatile ROM and I/O ports is also being developed to spread computation.

The AlgoBlock system [13] is a set of physical blocks that can be connected to each other to form a program. Each block corresponds to a single Logo-like command in the programming language. Once again, the emphasis is on manipulating physical blocks each with a designated atomic function which can be linked together to compose a more complex program. The system facilitates collaboration by providing simultaneous access and mutual monitoring of each block.

Based on a similar philosophy of the 3-Draw computer-aided design tool [11], Hinckley et al. has developed passive real-world interface props [5]. Here users are given physical props as a mechanism to manipulate 3D models. They are striving for interfaces in which the computer passively observes a natural user dialog in the real world (manipulating physical objects), rather than forcing a user to engage in a contrived dialog in the computer generated world.

Finally, the DigitalDesk [15] merges our everyday physical desktop with paper documents and electronic documents. A computer display is projected down onto a real physical desk and video cameras pointed at the desk use image analysis techniques to sense what the user is doing. The DigitalDesk is a great example of how well we can merge physical and electronic artifacts, taking advantage of the strengths of both mediums.

STAGE 1: EARLY BRICK EXPLORATIONS

A series of quick studies was conducted to motivate and investigate some of the concepts behind Graspable UIs. Having decided on bricks, we wanted to gain insights into the motor-action vocabulary for manipulating them.

LEGO separation task

The first exploratory study asked subjects to perform a simple sorting task as quickly as possible. The basic idea was to get a sense of the performance characteristics and a range of behavior people exhibit while performing a task that warrants rapid hand movements and agile finger control for object manipulation. Subjects were presented with a large pile of colored LEGO bricks on a table and were asked to separate them into piles by color as quickly as possible.

We observed rapid hand movements and a high degree of parallelism in terms of the use of two hands throughout the task. A very rich gestural vocabulary was exhibited. For instance, a subject's hands and arms would cross during the task. Subjects would sometimes slide instead of pick-up and drop the bricks. Multiple bricks were moved at the same time. Occasionally a hand was used as a "bulldozer" to form groups or to move a set of bricks at the same time. The task allowed subjects to perform *imprecise* actions and interactions. That is, they could use mostly ballistic actions throughout the task and the system allowed for imprecise and incomplete specifications (e.g., "put this brick in that pile," which does not require a precise (x, y) position specification). Finally, we noticed that users would enlarge their workspace to be roughly the range of their arms' reach.

Domino sorting task

The second exploratory study asked subjects to place dominos on a sheet of paper in descending sorted order. Initially, the dominos were randomly placed on a tabletop and subjects could use the entire work surface. A second condition was run which had the dominos start in a bag. In addition, their tabletop workspace was restricted to the size of a piece of paper.

Once again this sorting task also revealed interesting interaction properties. Tactile feedback was often used to grab dominos while visually attending to other tasks. The non-dominant hand was often used to reposition and align the dominos into their final resting place while, in parallel, the dominant hand was used to retrieve new dominos. The most interesting observation was that subjects seemed to inherently know the geometric properties of the bricks and made use of this everyday knowledge in their interactions without prompting. For example, if 5 bricks are side-by-side in a row, subjects knew that applying simultaneous pressure to the left-most and right-most end bricks will cause the entire row of bricks to be moved. Finally, in the restricted workspace domino condition we observed one subject taking advantage of the "stackability" of the dominos and occasionally piled similar dominos on top of others to conserve space. Also, sometimes a subject would use their non-dominant hand as a "clipboard" or temporary buffer while they plan or manipulate other dominos.

Physical manipulation of a stretchable square

To get a better sense of the issues for manipulating physical versus virtual objects, we designed a "stretchable square" constructed out of foam core. This square looks like a tray with a one inch rim around each side. Users could expand or collapse the width of the square (see Figure 7). We displayed an end position, orientation and scale factor for the physical square and asked subjects to manipulate the square to match the final target as quickly as possible. A variety of cases were tested involving one, two or all three transformation operations (translate, scale, and rotate).

FIGURE 7. Flexible curve and stretchable square.

We found that each subject had a different style of grasping the stretchable square for position and orientation tasks. This served to remind us that physical objects often have a wide variety of ways to grasp and to manipulate them even given natural grasp points. In addition, subjects did not hesitate and were not confounded by trying to plan a grasp strategy. One subject used his dominant hand to perform the primary manipulation and the non-dominant hand as a breaking mechanism and for finer control.

Perhaps the most salient observation is that users performed the three operations (translation, rotation and scaling) in parallel. That is, as the subjects were translating the square towards its final position, they would also rotate and scale the square at the same time. These atomic operations are combined and chunked together [1].

Comparison Using MacDraw Application

The same matching tasks were then done using virtual objects and a stylus on a large, horizontal drafting table with a computer display projected on the writing surface. Using the MacDraw II(TM) program, subjects were asked to move a virtual object on top of a target virtual object matching position, orientation and scale factors.

We observed that even when we factor out the time needed to switch in and out of rotation mode in MacDraw, task completion time was about an order of magnitude longer than the physical manipulation using the stretchable square. We noticed a "zoom-in" effect to reach the desired end target goal. For example, subjects would first move the object on top of the target. Then they would rotate the object, but often be unable to plan ahead and realize that the center of rotation will cause the object to be displaced. Thus, they often had to perform another translation operation. They would repeat this process until satisfied with a final match.

The MacDraw user interface, and many other interfaces, forces the subject to perform the operations in a strictly sequential manner. While we can become very adept at performing a series of atomic operations in sequence, the interface constrains user interaction behavior. In effect, the interface forces users to remain novices by not allowing them to exhibit more natural and efficient expressions of specifying atomic operations in parallel.

Curve Matching

Continuing to explore our skills at physical manipulations, we asked subjects to use a flexible curve (see Figure 7) to match a target shape. The flexible curve, often used in graphic design, consists of a malleable metal surrounded by soft plastic in the shape of a long (18 inch) rod. The inner metal allows the curve to hold its shape once deformed.

We found that users quickly learned and explored the physical properties of the flexible curve and exhibited very expert performance in under a minute. All ten fingers were often used to impart forces and counterforces onto the curve. The palm of the hand was also used to preserve portions of the shape during the curve matching task. We observed that some subjects would "semantically load" their hands and arms before making contact with the flexible curve in anticipation of their interactions. The semantic loading is a preconceived grasp and manipulation strategy by the user which, in order to execute properly, the arms, hands and fingers must start in a specific, sometime uncomfortable, loaded position. This process often allowed the subject to reach the final target curve shape in one gestural action.

STAGE 2: MOCK-UP AND SIMULATIONS

Next, we mocked-up some sample brick interactions using a prototyping tool (Macromind Director) and acted them out on the Active Desk. By using a few LEGO bricks as props and creating some basic animations using the prototyping tool, we could quickly visualize what the interactions would look like and begin to get a sense of how they will feel. These sample interactions were video taped and edited. We were able to mock-up many of the primary ideas such as: attaching and detaching bricks from virtual objects; translation and rotation operations using one brick; using two bricks each attached to separate virtual objects, and finally two bricks attached to a single virtual object to specify stretching and simple deformations.

All of these exploratory studies and mock-ups aided us to quickly explore some of the core concepts with minimum set-up effort. Finally, the video tapes that we create often serves as inspirational material.

STAGE 3: PROTOTYPE

After the mock-ups, we built the bricks prototype to further investigate the Graspable UI concepts. The prototype consists of the Active Desk, a SGI Indigo2 and two Ascension Bird receivers (see Figure 8).

Active Desk

The Active Desk is a large horizontal desktop surface which has a rear projected computer screen underneath the writing surface (see Figure 8). Modeled after a drafting table, the dimensions of the surface are roughly 4.5' by 3.0' on a slight 30 degree angle. The projected computer screen inset has a dimension roughly 3' by 2'. A Scriptel transparent digitizing tablet lays on top of the surface and a stylus device may be used for input. The LCD projection display only has a 640x480 resolution so the SGI screen is down converted to an NTSC signal and sent to the LCD display.

Bricks

To prototype the graspable objects (bricks), we use the Ascension Flock of Birds(TM) 6D input devices to simulate the graspable objects. That is, each receiver is a small 1 inch cube that constantly sends positional (x, y, and z) and orientation information to the SGI workstation. We currently have a two receiver system, which simulates two active bricks that operate on top of the Active Desk. More receivers can be added to the system but the wires attached to the receivers hinder interactions. Nevertheless, the two receivers offer us an initial means of exploring the design space in a more formal manner.

GraspDraw -- A simple drawing application

A simple drawing application, GraspDraw, was developed to test out some of the interaction techniques. The

application lets users create objects such as lines, circles, rectangles and triangles (see Figure 8). Once created, the objects can be moved, rotated and scaled. GraspDraw is written in C using the GL library on an SGI Indigo2.

The two Bird receivers act like bricks and can be used simultaneously to perform operations in parallel. One of the bricks has a push button attached to it to register additional user input. This button is primarily used for creating new objects. Grasps (i.e., attaching the brick to a virtual object) are registered when a brick is near or on the desktop surface. To release a grasp, the user lifts the brick off of the desktop (about 2 cm).

To select the current tool (select, delete, rectangle, triangle, line, circle) and current draw color, we use a physical tray and an ink-well metaphor. Users dunk a brick in a compartment in the tray to select a particular tool. A soft audio beep is heard to act as feedback for switching tools. Once a tool is selected, a prototype shape or tool icon is attached to the brick. The shape or icon is drawn in a semi-transparent layer so that users may see through the tool.

FIGURE 8. GraspDraw application and ActiveDesk.

The concept of an *anchor* and *actuator* have been defined in interactions that involve two or more bricks. An anchor serves as the origin of an interaction operation. Anchors often specify an orientation value as well as a positional value. Actuators only specify positional values and operate within a frame of reference defined by an anchor. For example, performing a stretching operation on a virtual object involves using two bricks one as an anchor and the other as an actuator. The *first* brick attached to the virtual object acts as an anchor. The object can be moved or rotated. When the *second* brick is attached, it serves as an actuator. Position information is registered relative to the anchor brick. If the first anchor brick is released, the actuator brick is promoted to the role of an anchor.

STAGE 4: COMMERCIAL APPLICATION

In following the goals of user centered design and user testing, there are some real problems when working with new interaction techniques such as the Graspable UI. First, in order to conduct formal experiments, one must generally work in a restricted controlled environment. The demands of experimental control are often at odds with human performance in the more complex context of the real world. Secondly, University researchers typically do not have access to the source code of anything but toy applications. Therefore, testing and demonstrations of innovative techniques like the Graspable UI are subject to criticisms that "it's fine in the simple test environment, but it won't work in the real world."

The first point can be dealt with by careful experimental design and differentiating between controlled experiments and user testing. Our approach to the second is to partner with a commercial software company that has a real application with real users. In so doing, we were able to access both a real application and a highly trained user community.

Hence, we have implemented a critical mass of the Graspable UI into a modified version of Alias Studio(TM), a high-end 3D modeling and animation program for SGI machines. Specifically, we are exploring how multiple bricks can be used to aid curve editing tasks. Although we have just begun this stage of research, we currently have two bricks integrated into the Studio program. The bricks can be used to simultaneously edit the position, orientation and scale factor for points along a curve. Future investigations may use bricks to clamp or freeze portions of the curve. This integration process and evaluation will further help us to refine the Graspable UI concepts.

DISCUSSION

We have conducted some preliminary user testing of the bricks concept using the GraspDraw application. All of the approximately 20 users who have tried the interface perform parallel operations (e.g., translate and rotate) at a very early stage of using the application. Within a few minutes of using the application, users become very adept at making drawings and manipulating virtual objects. Some users commented on the fact that the bricks were tethered, which hindered some of their interactions.

One could argue that all Graphical UI interactions, except perhaps touch (e.g., touchscreens) are already graspable interfaces if they use a mouse or stylus. However, this claim misses a few important distinctions. First, Graspable UIs

make a distinction between "attachment" and "selection." In traditional Graphical UIs, the selection paradigm dictates that there is typically only one active selection; Selection N implicitly causes Selection N-1 to be unselected. In contrast, when bricks are attached to virtual objects the association persists across multiple interactions. Selections are then made by making physical contact with the bricks. Therefore, with Graspable UIs we can possibly eliminate many of the redundant selection actions and make selections easier by replacing the act of precisely positioning a cursor over a small target with the act of grabbing a brick. Secondly, Graspable UIs advocate using multiple devices (e.g., bricks) instead of channeling all interactions through one device (e.g., mouse). Consequently, not only are selections persistent, there can be one persistent selection per brick. Thirdly, the bricks are inherently spatial. For example, we can temporarily arrange bricks to form spatial caches or use them as spatial landmarks for storage. By having more spatial persistence, we can use more of our spatial reasoning skills and muscle memory. This was exhibited during the LEGO and Domino exploratory studies. Clearly, the bricks are handled differently than a mouse.

One may suggest to eliminate using bricks and instead use only our hands as the physical input devices. While this may be useful for some applications, in general using a physical intermediary (i.e., brick) may be more desirable. First, tactile feedback is essential; it provides a way of safeguarding user intent. The bricks supply tactile confirmation and serve as a visual interaction residue. Secondly, hand gestures lack very natural delimiters for starting and stopping points. This makes it difficult to segment commands and introduces lexical pragmatics. In contrast, the affordances of touching and releasing a brick serve as very natural start and stop points.

There are many open design issues and interaction pragmatics to research. For example, should we vary the attributes of a brick (shape, size, color, weight) to indicate its function? Should all the bricks have symmetrical behavior? How many bricks can a user operate with at the same time? Do the bricks take up too much space and cause screen clutter (perhaps we can stack the bricks and they can be made out of translucent material)? For fine, precise pointing, do bricks have a natural hot spot (perhaps a corner or edge)? Sometimes it is more advantageous to have a big "cursor" to acquire a small target [6].

Inter-Brick behaviors

Much of the power behind the Bricks is the ability to operate and interact with more than one brick at the same time. Our interaction techniques need to be sensitive to this issue and define consistent inter-brick behaviors for one-handed (unimanual) or two-handed (bimanual) interactions. Moreover, we will need to develop a new class of techniques that use combinations of unimanual and bimanual interactions during the life span of a single technique. For instance, a technique may be initiated using one hand, transfer to using both hands and then terminate back to using one hand. The key point is that we need to provide for seamless transitions within a single interaction technique that switches between unimanual and bimanual interactions. As we noted earlier, the Anchor/Actuator behavior serves as one example.

Our goal has been to quickly explore the new design space and identify major landmarks and issues rather than quantify any specific subset of the terrain. The next phase of our evaluation will include a more detailed evaluation at places in the design space that have the most potential.

DESIGN SPACE

We have developed an initial design space for bricks which serves to lay the foundation for exploring Graspable UIs. Table 1 summarizes the design space. The shaded region in the table represents where our current Bricks prototype fits in the design space. Each of the rows in the table represent dimensions of the design space which are described below.

TABLE 1. Design space of Bricks for Graspable User Interfaces. Gray region shows where current Brick prototype fits into design space.

Brick's internal ability -- Does the brick have any internal mechanisms (physical or electronics) that generates additional information or intelligence? Inert objects have no internal mechanisms, only external features (color, shape, weight). Smart objects often have embedded sensors and microprocessors.

Input & Output -- What properties can be sensed and displayed back to the user (or system)?

Spatially aware -- Can the brick sense the surroundings and other bricks? Bricks can be unaware (work in isolation); mutually aware (aware only of each other); or be aware of their surroundings (primitive sensing of its environment and other bricks) [4].

Communication -- How do the bricks communicate among themselves and to host computers? The mechanisms range from wireless (such as infra-red), tethered (requiring wires or cables) and grid board (a specialized operating surface with pluggable connecting parts).

Interaction time span -- Given a task, are users manipulating the bricks in quick bursty interactions (sometimes gesturing in fractions of a second); using a set of bricks, accessing them within seconds or minutes (an interaction cache); or are the interactions long term running days, months, years between interactions (e.g., an archive)?

Bricks in use at same time -- Do users manipulate one brick at a time (one handed interactions), two at a time (two handed interactions), or more than two? Users could manipulate 5 to 10 bricks at a time (e.g., bulldozer) or perhaps even 50 to 100 at a time.

Function assignment -- How frequently and by what mechanism do the bricks get assigned their functions? Permanent assignment means that each brick has one function or role for its lifetime. With some effort, programmable assignment allows bricks to have their function reassigned. Transient assignment allows for users to rapidly reassign the brick's function.

Interaction representations -- Is the system designed to have a blend of physical and virtual artifacts? When there is a mix, are the dual representations equal (i.e., functions can be performed using either physical or virtual artifacts), complimentary (i.e., one medium can perform a subset of the functionality that the other medium cannot) or combinatoric (together both offer functionality that either one could not provide alone).

Physical & Virtual layers -- Are the layers direct (superimposed) or indirect (separated)?

Bond Between Physical & Virtual layers -- Tightly coupled systems have the physical and virtual representations perfectly synchronized, the physical objects are tracked continuously in real time. Loosely coupled systems allow for the representations to become out of synchronization for long stretches of time (i.e., minutes, days) and updated in more of a batch mode.

Operating Granularity -- What is the range of space that the bricks operate in and at what sensing resolution? For example, the devices may operate at a *building* level (e.g., capable of determining what room they are currently in), *room* level (e.g., capable of determining, within an inch accuracy, position and orientation information inside a room), and *desktop* level (e.g., micro accuracy within 0.1 in of position and orientation information on a desktop).

Operating surface texture -- What granularity or texture do the bricks operate on? A discrete texture requires that the bricks be plugged into special receptors (e.g., a grid board) while a continuous texture allows for smooth movement or dragging (e.g., tabletop).

Operating surface type -- Do the bricks operate on a static surface (e.g., a tabletop) or a dynamic surface which can be changing constantly (e.g., Active Desk)?

It should be noted that this is not an exhaustive parsing of the design space. Robinett [10], however, proposes a more formal taxonomy for technologically mediated experiences which may aid our investigation. Yet, the many dimensions of our design space exhibit its richness and provides a more structured mechanism to explore the concepts behind Graspable UIs.

FUTURE WORK

There are many future directions we would like to explore. First, we will conduct more formal evaluation measures on the GraspDraw program. Next we will investigate other regions of the design space including developing techniques in 3-D as well as to operate on 3-D virtual objects. In addition, we hope to develop multiple, untethered bricks. Two promising areas are computer vision techniques [12] and electric-field sensing [16].

CONCLUSIONS

In this paper we have introduced a new technique, the Graspable User Interface, as a means of augmenting the power of conventional Graphical User Interfaces. In so doing, we have attempted to go beyond a simple "show and tell" exercise. Through the methodology described, we have attempted to both explore the overall design space effectively, and tease out the underlying human skills on which we could build our interaction techniques.

The Graspable User Interface is an example of "radical evolution." It is evolutionary in the sense that it builds upon the conventions of the GUI. Hence, both existing technology and human skill will transfer to the new technique. However, it is radical in that the incremental change that it introduces takes us into a radically new design space. Assuming that this new space is an improvement on what preceded it, this combination gives us the best of both worlds: the new and the status quo.

From the experience gained in the work described, we believe these new techniques to be highly potent and worthy of deeper study. What we have attempted is a proof of concept and exposition of our ideas. Hopefully this work will lead to a more detailed exploration of the technique and its potential.

Acknowledgments

This research was undertaken under the auspices of the Input Research Group at the University of Toronto and we thank the members for their input. We especially would like to thank Mark Chignell, Marilyn Mantei, Michiel van de Panne, Gordon Kurtenbach, Beverly Harrison, William Hunt and Kim Vicente for their input. Thanks also to Ferdie Poblete who helped design and build the stretchable square prop. The Active Desk was designed and built by the Ontario Telepresence Project and the Arnott Design Group. Our research has been supported by the Information Technology Research Centre of Ontario, the Natural Sciences and Engineering Research Council of Canada, Xerox PARC and Alias Research.

More material including dynamic figures can be found on the CHI'95 Electronic Proceedings CD-ROM and at URL: http: //www.dgp.utoronto.ca/people/GeorgeFitzmaurice/home.html

DYNAMIC FIGURE 1 (QuickTime Movie, about 10 mb) **No caption given.**

References

1. Buxton, W. (1986). Chunking and phrasing and the design of human-computer dialogues. *Proceedings of the IFIP World Computer Congress*. pp. 475-480.
2. Eilan, N., McCarthy, R. and Brewer, B. (1993). Spatial Representation. Oxford, UK: Blackwell.
3. Guiard, Y. (1987). Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain Model. In *Journal of Motor Behavior*, 19(4), pp. 486-517.
4. Fitzmaurice, G.W. (1993). Situated Information Spaces and Spatially Aware Palmtop Computers, *Communications of the ACM*. 36(7), pp. 38-49.
5. Hickley, K., Pausch, R., Goble, J. C. and Kassell, N. F. (1994). Passive Real-World Interface Props for Neurosurgical Visualization. *Proc. of CHI'94*, pp. 452-458.
6. Kabbash, P. and Buxton, W. (1995). The 'Prince' Technique: Fitts' Law and Selection Using Area Cursors, To appear in *Proc. of CHI'95*.
7. Kabbash, P., Buxton, W. and Sellen, A. (1994). Two-Handed Input in a Compound Task. *Proc. of CHI94*, pp.

417-423.
8. MacKenzie, C. L. and Iberall, T. (1994). The Grasping Hand. Amsterdam: North-Holland, Elsevier Science.
9. Resnick, M. (1993). Behavior Construction Kits. In *Communications of the ACM.* 36(7), pp. 64-71.
10. Robinett, W. (1992). Synthetic Experience: A Proposed Taxonomy. *Presence*, 1(2), pp. 229-247.
11. Sachs, E., Roberts, A. and Stoops, D. (1990). 3-Draw: A tool for the conceptual design of three-dimensional shapes. CHI'90 Technical Video Program, ACM SIGGRAPH Video Review, Issue 55, No. 2.
12. Schneider, S.A. (1990). Experiments in the dynamic and strategic control of cooperating manipulators. Ph.D. Thesis, Dept. of Elec. Eng., Stanford Univ.
13. Suzuki, H., Kato, H. (1993). AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning. Proceedings of 4th European Logo Conference, Aug. 1993, Athens Greece, pp. 297-303.
14. Weiser, M. (1991). The computer for the 21st Century. In *Scientific America*, 265(3), pp. 94-104.
15. Wellner, P. (1993). Interacting with paper on the DigitalDesk. In *Com. of the ACM.* 36(7), pp. 86-96.
16. Zimmerman, T., Smith, J.R., Paradiso, J.A., Allport, D. and Gershenfeld, N. (1995). Applying Electric Field Sensing to Human-Computer Interfaces. To Appear in *Proceedings of CHI'95.*

BUILD-IT: A Planning Tool for Construction and Design

Matthias Rauterberg
Morten Fjeld
Helmut Krueger

Institute for Hygiene and Applied Physiology (IHA)
Swiss Federal Institute of Technology (ETH)
Clausiusstrasse 25, CH-8092 Zurich,
SWITZERLAND
rauterberg@iha.bepr.ethz.ch

Martin Bichsel
Uwe Leonhardt
Markus Meier

Institute for Construction and Design (IKB)
Swiss Federal Institute of Technology (ETH)
Tannenstr. 3, CH-8092 Zurich,
SWITZERLAND
mbichsel@ikb.mavt.ethz.ch

ABSTRACT

It is time to go beyond the established approaches in human-computer interaction. With the Augmented Reality (AR) design strategy humans are able to behave as much as possible in a natural way: behavior of humans in the real world with other humans and/or real world objects. Following the fundamental constraints of natural way of interacting we derive a set of recommendations for the next generation of user interfaces: the *Natural User Interface* (NUI). The concept of NUI is presented in form of a runnable demonstrator: a computer vision-based interaction technique for a planning tool for construction and design tasks.

Keywords

augmented reality, digital desk, natural user interface, computer vision-based interaction

INTRODUCTION

The embodiment of computers in the work place has had a tremendous impact on the field of human-computer interaction. Mouse and graphic displays are everywhere, the desktop workstations define the frontier between the computer world and the real world. We spend a lot of time and energy to transfer information between those two worlds. This could be reduced by better integrating the virtual world of the computer with the real world of the user. Following the argumentation of Fitzmaurice, Ishii and Buxton [3] a graspable user interface has the following advantages: "• It encourages two handed interactions; • shifts to more specialised, context sensitive input devices; • allows for more parallel input specification by the user, ... • leverages off of our well developed skills ... for physical object manipulations; • externalises traditionally internal computer representations; • facilitates interactions by making interface elements more 'direct' and more 'manipulable' by using physical artifacts; ... • affords multi-person, collaborative use" ([3] p.443).
Summarising we can conclude the following design recommendations: To empower the human to computer interaction, the user must be able to behave in a *natural way* bringing into action all of his or her body parts (e.g., hands, arms, face, head, voice, etc.). To interpret all of these natural expressions we need very powerful and intelligent pattern recognition techniques.

THE DEMONSTRATOR "BUILD-IT"

Inspired by the ideas of Tognazzini [5], the SUN StarFire video, and Wellner's digital desk [6] we designed a system that is based primarily on the concept of Natural User Interfaces (NUIs, see [4]). We choose the task context of planning activities for plant design. A system (patent pending), called "BUILD-IT", has been realized and an application, supporting engineers in designing assembly lines and building plants, was implemented. The realized design room (see Figure 1) enables the users to sit around a table and to act in the interaction space (the top view) with a mixture of virtual and real world objects on the table. The vertical area (the side view in the background of Figure 1) is used for a 3D perspective into the designed plant.

Figure 1: The design table of BUILD-IT with the 2D top view on the table, and the 3D side view in the back.

The BUILD-IT hardware has five components:
1) A table with a white surface is used as a horizontal projection and interaction area (the top view). An ASK 960 high resolution LCD projector projects the top view of the 2D computer graphics scene vertically down to the table.

2) A white projection screen provides a vertical projection area (the side view). An ASK 860 high resolution LCD projector projects the 3D perspective view into a computer graphics scene horizontally onto the projection screen.
3) A CCD camera with a resolution of 752 (H) by 582 (V) pixels looks vertically down to the table.
4) A small brick with a special surface is the physical 'interaction handler' (replacement for the 'old' mouse cursor).
5) A low-cost Silicon Graphics Indy (IP22 133 MHz processor, R4600 processor, and the standard Audio-Video Board) provides the computing power for digitising the video signal coming from the camera, analysing the users' interactions on the table, and rendering the interaction result in the two views.

The BUILD-IT software consists of two independent processes communicating with each other through socket connection.

1) The real time analysis process of the video reads images from the camera, extracts contours of moving objects, analyses these contours [2], and determines the position and orientation of the universal interaction handler (the brick).
2) An application is built on top of the multi-media framework MET++ [1]. This application interprets the user action based on the position and orientation of the interaction handler, modifies a virtual scene according to the user action, and renders the top view of the new scene (for the vertical projector) and the side view into the scene (for the horizontal projector).

The application is designed for supporting providers of assembly lines and plants in the early design processes. It can read and render arbitrary CAD models of machines in VRML format. The input of a 3D model of the virtual objects is realized by connecting BUILD-IT with the CAD-System CATIA. So it was possible to import the original CAD-models into BUILD-IT.

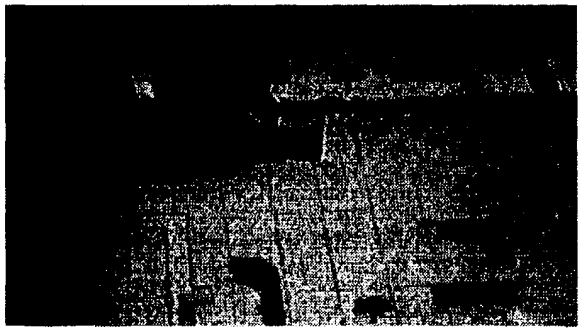

Figure 2: The object menu (white area, the 'virtual machine store'), the top view (grey), and the hand moving the brick.

BUILD-IT currently supports the following user interactions: selection (Figure 3a), positioning (Figure 2), deselection (Figure 3b), reselection, deleting, and moving of a virtual camera.

Selecting of a virtual object (e.g., a specific machine) in a 'virtual machine store' is done by placing the interaction handler onto the projected image of the machine in the menu area on the table.

Positioning a machine in the virtual plant by moving the interaction handler to the corresponding position in the projected plant layout on the table. Positioning includes machine orientation that is coupled to the orientation of the interaction handler.

Deselecting and fixing the machine by covering the surface of the interaction handler with the hand and removing it.

Re-selection of a machine by placing the interaction handler onto the specific machine.

Deleting the machine by moving it back into menu area (the 'virtual machine store', see Figure 2).

Moving a virtual camera through the plant, rendering the perspective view of this camera, and displaying it on the vertical projection screen (the side view, see Figure 1).

Figure 3: (a) selection, and (b) deselection of an object.

To change the side view the virtual camera can be moved on a level with the eyes of a virtual person of 1.7 m body height. The virtual camera is selected and moved exactly the same way as all other virtual objects. The current view position and viewing direction corresponds to the current position and orientation of the virtual camera icon in the plant layout. The system has been empirically tested with several managers and engineers out of companies that produce assembly lines and plants. These tests showed that the system is intuitive to use, easy to learn, and that people could enjoy using it. Most persons were able to assemble virtual plants after only 30 seconds of introduction to the system.

REFERENCES

1. Ackermann, P. *Developing Object-Oriented Multimedia Software Based on the MET++ Application Framework*. Heidelberg: dpunkt, 1996.
2. Bichsel, M. Segmenting Simply Connected Moving Objects in a Static Scene. *Transactions on Pattern Recognition and Machine Intelligence (PAMI)*, Vol. 16, No. 11, Nov. 1994, pp. 1138-1142.
3. Fitzmaurice, G., Ishii, H., and Buxton, W. Bricks: Laying the Foundations for Graspable User Interfaces. In *Proc. of the CHI '95*, 1995, pp. 442-449.
4. Rauterberg, M., and Steiger, P. Pattern recognition as a key technology for the next generation of user interfaces. In *Proc. of IEEE--SMC'96* (Vol. 4, 1996, pp. 2805-2810). Piscataway.
5. Tognazzini, B. *Tog on Software Design*. Addison-Wesley, Reading MA, 1996.
6. Wellner, P. Interacting with Paper on the Digital Desk. *Communications of the ACM*, 36(7), 1993, pp. 87-96.